«ИНФОРКОМ»

MMIIRR LANDS

Circle Lebedev

1995

PDF version by Deny (Deniseno D.A.)
e-mail: DenyDA@mail.ru
2009

ENTERPRISE FROM PDF'ika.
(DNY)

A lot of people who read these lines think where this book came from, didn't they? And You'll be right. Partly.

At the beginning, I downloaded txt file with Virtual TR-DOS, on the news for 11.03.2006. Figures below:

@ Another update under Books. Okay, Oleg Dagtyre sent a book of fascinating games. On a domestic computer, and Slave Kalinin dug a book of the World of Sounds. Spectrum. The author of the last book on his website claims that the manuscript was in time. Ambassador Inforcoma, but he never saw the light.

The author's wrong. The book was printed, but electronically in ZX-FORUM No. 4, really, no. A picture. I've decided to make this book in print, that's PDF. Total 2 years of work (more abandoned) You'll see for yourself.

Deny (DenyDA@mail.ru)

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Эта книга предназначена для читателей, которые решили писать программы, и хотят оформить их на высоком уровне, что невозможно без использования звуковых возможностей ZX-Spectrum. Она составлена по принципу от простого - к сложному, и разобраться в ней сможет даже не очень подготовленный человек. Также книга может служить пособием для начинающих программировать на ассемблере: она буквально напичкана примерами.

Здесь содержатся практически все сведения, которые могут Вам понадобиться когда-нибудь для создания звукового сопровождения как игровых, так и системных программ.

В книге рассматривается программирование только стандартных звуковых устройств Спектрума (порт #FE и музыкальный сопроцессор). Такая экзотика, как ЦАП, АЦП, MIDI, присущая Спектрумам - монстрам (ATM TURBO, PROFI) и некоторым самоделкам, даже не затрагивается из-за её нестандартности. Хотя, с помощью одного только ЦАПа (Цифро-Аналогового Преобразователя) можно было бы создавать почти фантастические звуки. Возможно, когда-нибудь в будущем, я опишу и эти средства создания звука.

Теперь о том, что же здесь все-таки содержится.

В первой главе рассказывается о природе звука и о его характеристиках.

В следующих трех главах описывается оператор Spectrum-бейсика BEEP, создание с его помощью эффектов и мелодий.

В следующей главе рассказывается о том, как в ZX-Spectrum получается звук, а также о порте 254(#FE).

Дальше повествуется о создании эффектов и музыки в машинных кодах.

Затем - об обработке сигналов, подаваемых на магнитофонный вход Вашего компьютера, и о программах, синтезирующих человеческую речь.

В следующей главе рассказывается о звуке на прерываниях.

Вторая часть книги посвящена музыкальному сопроцессору AY-3-8910 (AY-3-8912).

Сначала описывается оператор БЕЙСИКА-128 PLAY и всё, что с ним связано.

Следующая глава поможет Вам разобраться в управлении сопроцессором и в его регистрах. С её помощью Вы научитесь программировать микросхему AY в кодах.

И, наконец - обзор программного обеспечения, в который включены различные музыкальные редакторы, редакторы звуковых эффектов, синтезаторы речи, демонстрационные программы и т.п. Подробно описаны три такие программы. Это SUPER SOUND - редактор звуковых эффектов, WHAM THE MUSIC BOX - музыкальный редактор и ASC SOUND MASTER (ASM) -музыкальный редактор для сопроцессора.

В приложении 1 находятся листинги всех звуковых эффектов программы SUPER SOUND, а в приложении 2 даны советы по использованию ассемблера GENS4.

Условные обозначения, используемые в книге:

CS - клавиша Caps Shift;
SS - клавиша Symbol Shift;
CS/1 - одновременное нажатие клавиш (в данном случае Caps Shift и 1);
A+2 - последовательное нажатие клавиш (в данном случае A и 2);
[1] - номер издания в списке литературы, в конце книги.

1. Физика звука.

Говоря научным языком, звук - это механические продольные колебания упругой среды. Выражаясь же языком простых смертных - это всего-навсего колебания воздуха.

Человеческое ухо очень чувствительно к изменению воздушного давления, оно улавливает малейшие его колебания.

Человек может воспринимать звуковые колебания с частотой от 20 до 20000 Гц (Гц - герц. Одно колебание в секунду). Звук с частотой меньше 20 Гц называется инфразвуком, а больше 20000 Гц - ультразвуком.

Чем больше частота колебаний, тем выше слышимый звук. Чувствительность уха не одинакова на всем диапазоне звуковых частот. Самые низкие и самые высокие частоты воспринимаются гораздо хуже средних.

Громкость звука напрямую зависит от амплитуды колебаний и измеряется в децибелах (дБ). Чем больше амплитуда, тем громче слышимый звук. Человек может воспринимать звук с громкостью от 0 до 130 дБ. Самые громкие звуки (100 - 130 дБ) воспринимаются болезненно, а при громкости больше 130 дБ лопаются барабанные перепонки в ушах и человек становится глухим.

Следующая характеристика звука - тембр. Тембр - это форма звуковой волны. Благодаря тому, что наше ухо способно различать тембры, мы можем отличать звучание гитары от фортепиано. Для наглядности взгляните на рис. 1. Там приведены две звуковые волны с различными тембрами.

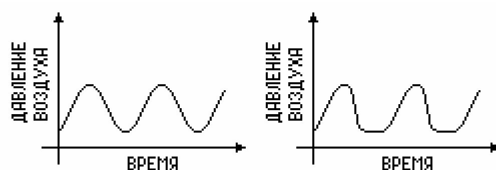


Рис. 1 Звуковые тембры.

Ещё одна характеристика - периодичность. Если звуковая волна состоит из одного повторяющегося фрагмента, то такой звук можно считать периодичным, а в противном случае - нет. Периодичный звук воспринимается как музыкальный тон, а непериодичный - как шум. Существует несколько различных видов шума. Два наиболее распространенных - белый шум и импульсный шум. Их диаграммы приведены на рис. 2.

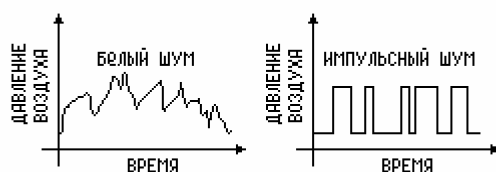


Рис. 2 Белый и импульсный шумы.

Последняя характеристика звука - длительность. Она создана искусственно, так как определяет не сам звук, а время его звучания. В основном эта характеристика используется в музыке.

Все перечисленные выше параметры звука довольно относительны, потому что каждый человек ощущает их по-разному. Например, один и тот же звук может одному человеку казаться слишком громким, а другому - слишком тихим.

2. Оператор BEEP.

Единственная возможность получения звука, которую предоставляет Spectrum- бейсик - это оператор BEEP. Его формат:

```
BEEP t,f
```

- где t - длительность звука в секундах, а f - высота в полутонах. Чтобы ноту повысить или понизить на октаву необходимо её значение соответственно увеличить, или уменьшить на 12. Так

```
BEEP 0.5,0
```

воспроизведет ноту ДО первой октавы в течение полсекунды, а

```
BEEP 0.5,12
```

- ту же ноту ДО, но уже второй октавы.

Указанное нотоисчисление не очень точно, поэтому взгляните на рис. 3. Там в верхней строчке приведены приблизительные, а в нижней - точные параметры оператора BEEP для первой октавы.

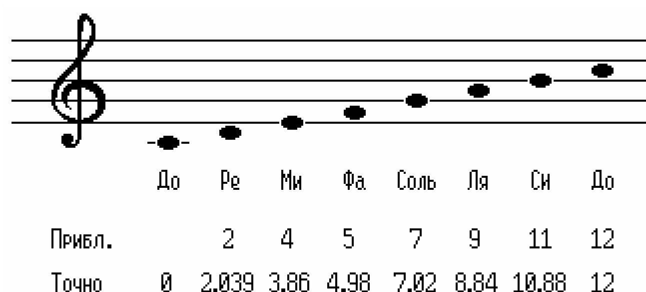


Рис. 3 Параметры BEEP для первой октавы.

Музыкальный диапазон BEEP достаточно широк - от -60 до 69. Он охватывает 18.5 октав, но самые низкие и самые высокие из них не представляют практического интереса для создания музыки, хотя вполне могут использоваться в звуковых эффектах.

И в заключение можно добавить об удобстве, которое Spectrum-бейсик предоставляет нам в записи десятичных дробей. Если такая дробь меньше единицы, то ноль перед точкой можно не ставить. Таким образом

```
BEEP 0.125,11
```

можно записать как

```
BEEP .125,11.
```

Дальше по тексту книги я именно так и буду делать.

2.1. Создание эффектов на BEEPe.

Для того чтобы сделать что-либо поинтереснее простых гудков, оператор BEEP обычно включается в какой-нибудь цикл, где изменяются один или оба его параметра. Простейший пример может выглядеть так:

```
10 FOR A=0 TO 60
20 BEEP .01,A
30 NEXT A
```

Эту маленькую программку можно долго и упорно изменять, получая все новые и новые эффекты. Попробуйте модифицировать пределы, шаг, направление цикла, длительность в операторе BEEP. Как это сделать, я думаю, понятно всем, поэтому примеров приводить не буду. Во время ваших экспериментов помните, что при маленьких длительностях низкие ноты воспроизводиться не будут, и чем ниже нота, тем больше должна быть длительность.

Когда Вам надоест занятие, предложенное в предыдущем абзаце, можно переходить к следующему этапу - изменению "внутренностей" цикла…

Можно увеличить число операторов BEEP. Причем, они могут быть настроены как на одинаковые частоты и длительности, так и на разные:

```
10 FOR A=0 TO 20
20 BEEP .007,A+7: BEEP .003,A
30 NEXT A
```

Диапазон звучания дополнительных BEEPов можно ограничить:

```
10 FOR A=0 TO 69
20 BEEP .01,A
30 IF A>20 AND A<40 THEN BEEP .01,A+8
40 NEXT A
```

или так:

```
10 FOR A=0 TO 69
20 BEEP .0003,A-6: BEEP .001,A-3: BEEP .01,A
30 NEXT A
```

Последний эффект интересен тем, что в нем второй и третий BEEPы вступают постепенно (см. сноску на предыдущей странице).

Эффект можно замедлить с помощью оператора PAUSE:

```
10 FOR A=10 TO 40 STEP 2
20 BEEP .01,A: PAUSE 2
30 NEXT A
```

Можно в течение воспроизведения менять длительность:

```
10 LET T=.01
20 FOR A=10 TO 60: BEEP T,A
30 LET T=T-.0002: NEXT A

10 FOR A=0 TO 60
20 BEEP A/500,60-A
30 NEXT A
```

В первом из этих примеров попробуйте поменять шаг или направление изменения длительности, начальное её значение.

Следующий вариант - вложенные циклы, которых может быть сколько угодно. BEEPы могут встречаться в любом из них:

```
10 FOR A=2 TO 5: BEEP .01,A
20 FOR B=20 TO 30: BEEP .01,B
30 NEXT B: NEXT A

10 FOR A=1 TO 40 STEP 10
20 FOR B=A TO A+20
30 BEEP .03,B
40 NEXT B: NEXT A
```

Попробуйте объединить несколько циклов. Например:

```
10 FOR A=12 TO 69: BEEP .001,A
20 NEXT A
30 FOR A=69 TO 12 STEP -1
40 BEEP.001,A: NEXTA
```

И, наконец, во всю эту коллекцию можно добавить элемент случайности:

```
10 FOR A=1 TO 10: LET F=RND*50
20 BEEP .1,F: BEEP .1,F+7: BEEP .1,F+4
30 NEXT A

10 FOR A=1 TO 20
20 BEEP .1,RND*20+40
30 NEXT A

10 FOR A=1 TO 20
20 BEEP RND/6,RND*60
30 NEXT A
```

Все перечисленные выше методы создания эффектов могут использоваться совместно, в любой комбинации.

Ищите, пробуйте, и Вы найдете среди множества парочку подходящих для Вашей программы эффектов.

2.2. Создание музыки на BEEPе.

К сожалению, на операторе BEEP невозможно создать сложную музыку, но наиграть

простенькую мелодию на нем можно.

Первое, что приходит в голову при попытке создать мелодию, это составить цепочку из BEEPов, которая обещает быть весьма неопределенных размеров. Этот способ примитивен, неудобен, занимает много места и вообще не очень красив. Ему существует несколько достойных замен. Например, можно использовать такой цикл:

```
10 FOR A=1 TO 10
20 READ T,F: BEEP T,F
30 NEXT A
```

И где-нибудь в программу вставить строку, начинающуюся с оператора DATA, и содержащую длительности и высоты нот мелодии. В данном примере их должно быть по десять.

В некоторых музыкальных произведениях бывают паузы. Чтобы исполнять такие мелодии, приведенный выше цикл придется немного модифицировать:

```
10 FOR A=1 TO 10: READ T,F
20 IF T=0 THEN PAUSE F*50: NEXT A
30 BEEP T,F: NEXT A
```

Теперь достаточно в данные вставить 0, а за ним длительность паузы в секундах. Вы, наверное, заметили, что значение паузы умножается на 50. Это делается потому, что параметр оператора PAUSE исчисляется в пятидесятых долях секунды.

В таблице 1 приведено соответствие нот длительностям операторов BEEP и PAUSE. В этой таблице даны стандартные значения нот и пауз, но в принципе можно использовать любые числа.

Вот, к примеру, программа, исполняющая куплет детской песенки "В лесу родилась елочка":

```
10 FOR A=1 TO 29: READ T,F
20 IF T=0 THEN PAUSE F*50: NEXT A
30 BEEP T,F: NEXT A
9000 DATA .25,7.02,.25,15.86,.25,15.86,.25,14.039,.25,15.86
9010 DATA .25,12,.25,7.02,.25,7.02,.25,7.02,.25,15.86,.25,15.86
9020 DATA .25,14.039,.25,20.84,.5,19.02,0,.5,.25,19.02,.25,8.84
9030 DATA .25,8.84,.25,16.98,.25,16.98,.25,15.86,.25,14.039,.25,12
9040 DATA .25,7.02,.25,15.86,.25,15.86,.25,14.039,.25,15.86,.5,12
```

В эту программу можно ввести возможность изменения темпа. Для этого необходимо первые три строки заменить следующими:

```
5 LET N=.5
10 FOR A=1 TO 29: READ T,F
20 IF T=0 THEN PAUSE F*N*50: NEXT A
30 BEEP T*N,F: NEXT A
```

Теперь Вы можете добиться желаемого темпа исполнения мелодии, подставляя различные значения в пятую строку. Однако не советую Вам вставлять туда 0 и очень большие числа.

Можно ещё усовершенствовать эту программку. Например, вставить контроль нажатия клавиши, сделать её более гибкой в отношении длины мелодии, оформить как подпрограмму и т. д. Конечный вариант может выглядеть так:

```
9090 DATA 100
9100 LET N=.5: RESTORE 9000
9110 READ T,F
9120 IF T=100 OR INKEY$<>"" THEN RETURN
9130 IF T=0 THEN PAUSE F*N*50: GO TO 9110
9140 BEEP T*N,F: GO TO 9110
```

Удалите все строки с 5 по 30 предыдущей программы и введите эти. Теперь Вы можете вызывать мелодию на исполнение оператором GO SUB 9100. Кроме того, теперь не надо старательно вычислять количество нот в мелодии: достаточно в её конец вставить число 100 (что собственно и сделано в строке 9090). Также Вы можете в любой момент прервать исполнение, нажав любую клавишу. Если мелодия кончится - подпрограмма также закончит свою работу. При повторном запуске мелодия начнется сначала, но, если убрать оператор RESTORE 9000, то она продолжится с того места, где была прервана (если, конечно, Вы не использовали оператор READ или RESTORE после её вызова).

Можно сделать проигрывание циклическим. Для этого придется поменять ещё пару строк:

```
9120 IF INKEY$<>"" THEN RETURN
9125 IF T=100 THEN GO TO 9100
```

Теперь вернуться из подпрограммы можно только по нажатию клавиши. Зато мелодия будет звучать пока кому-нибудь совсем не надоест.

Программы такого типа идеальны для простого исполнения, без замены и сохранения мелодии, но, если Вы хотите записать свое произведение на магнитофон, и загружать его прямо из работающей программы, то Вам подойдет следующий фрагмент:

```
10 FOR A=3 TO D(1)*2+2 STEP 2
20 IF D(A)=0 THEN PAUSE D(A+1)*D(2)*50: NEXT A
30 BEEP D(A)*D(2),D(A+1)
40 NEXT A
```

Для того чтобы он заработал, перед его запуском необходимо объявить и заполнить массив D, содержащий такую информацию о мелодии:

D(l) - количество нот
D(2) - темп исполнения
D(3) - длительность 1
D(4) - нота 1
.
.
.
D(N) - длительность N
D(N+l)-нота N

Его можно загрузить с кассеты, подав прямую команду

```
LOAD "name" DATA D()
```

или заполнить из строчек DATA, следующей программкой:

```
10 DIM D(20): RESTORE
20 FOR A=1 TO 20: READ D(A)
30 NEXT A
```

Обратите внимание, что первые две цифры данных должны быть количеством нот и темпом.

Созданный таким образом массив можно сохранить на ленте, командой

```
SAVE "name" DATA D()
```

Приведу ещё один пример подобной программы, оформленный по последнему писку моды:

```
9100 LET A=2
9110 IF D(A)=100 THEN GO TO 9100
9120 IF INKEY$<>"" THEN RETURN
9130 IF D(A)=0 THEN PAUSE D(A+1)*D(1)*50: GO TO 9150
9140 BEEP D(A)*D(1),D(A+1)
9150 LET A=A+2: GO TO 9110
```

Массив для этой подпрограммы должен быть таким:

D(l) - темп исполнения
D(2) - длительность 1
D(3) - нота 1
.
.
.
D(N) - длительность N
D(N+l)-нота N
D(N+2) - 100

Заметьте, что количество нот указывать не надо. На основе этого примера можно сделать даже музыкальный редактор.

Теперь вернемся к составлению цепочек данных.

Допустим, на нотном стане Вы увидели ноту точь в точь, как на рис. 4. Это ЛЯ первой октавы. Из рисунка 3 видно, что её частота запишется как 8.84.

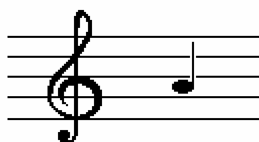


Рис. 4

Это одна четвертая, и, взглянув на таблицу 1, можно определить, что её длительность будет .25. Таким образом, пара данных для этой ноты будет .25,8.84. Для всех последующих нот вычисления будут такими же.

Нота		Зн. BEEP	Зн. PAUSE
Одна шестнадцатая	𝅘𝅥𝅯	.0625	3.125
Одна шестнадцатая с точкой	𝅘𝅥𝅯.	.09375	4.6875
Одна восьмая	𝅘𝅥𝅮	.125	6.25
Одна восьмая с точкой	𝅘𝅥𝅮.	.1875	9.375
Одна четвертая	𝅘𝅥	.25	12.5
Одна четвертая с точкой	𝅘𝅥.	.375	18.75
Одна вторая	𝅗𝅥	.5	25
Одна вторая с точкой	𝅗𝅥.	.75	37.5
Целая	𝅝	1.0	50

Табл. 1. Длительности нот для BEEP и PAUSE.

3. Как получается звук.

К сожалению, в ZX-Spectrum для управления звуком отведен всего-навсего один бит. Это бит D4 порта 254 (#FE). Но, несмотря на это, программисты умудряются создавать довольно красивые мелодии и эффекты.

Когда этот бит, установлен, на динамик (усилитель) подается напряжение, и его мембрана находится в одном положении. Когда же он сброшен, на динамике напряжения нет, и мембрана находится в другом положении. Таким образом, если Вы будете изменять состояние этого бита с достаточно высокой частотой, то мембрана будет вибрировать, и Вы услышите звук.

Использовать этот способ можно даже из бейсика:

```
10 FOR A=1 TO 300: OUT 254,23
20 OUT 254,7: NEXT A
```

И так же легко на ассемблере, но в этом случае придется запрещать прерывания, если Вы хотите получить качественное звучание:

```
10          DI                ; запрет прерываний
20          LD    BC,2560     ; BC=длительность
30          LD    A,7         ; A=цвет бордюра
40 BEGIN    XOR   16          ; инвертирование бита D4
50          OUT   (254),A     ; вывод A в порт 254
60          LD    D,100       ; D=задержка (частота)
70 PAUSE    DEC   D           ; D=D-1
80          JR    NZ,PAUSE    ; если D<>0, то цикл
90          DEC   BC          ; BC=BC-1
100         LD    D,A         ; сохранение A
110         LD    A,B         ; BC=
120         OR    C           ;      0 ?
130         LD    A,D         ; восстановление A
140         JR    NZ,BEGIN    ; если BC<>0, то цикл
150         EI                ; разрешение прерываний
160         RET               ; возврат в бейсик
```

Вы, наверное, заметили, что в этих программах в порт выводится не 16 и 0, что соответствовало бы установке и сбросу бита D4, а 23 и 7. Дело в том, что этот порт кроме динамика управляет ещё цветом бордюра и выходом на магнитофон. Познакомимся с его возможностями поближе:

биты D0…D2 определяют цвет бордюра:

000(0) - черный	100(4) - зеленый
001(1) - синий	101(5)- голубой
010(2) - красный	110(6) - желтый
011(3) - фиолетовый	111(7) - белый

бит D3 управляет выходом на магнитофон,

бит D4 управляет звуком,

биты D5…D7 не используются.

При вводе байта из порта бит D6 контролирует магнитофонный вход.

Думаю, теперь понятно, откуда взялись цифры 23 и 7: для установки и сброса бита D4 необходимо последовательно вывести в порт значения 16 и 0. Но при этом надо сохранять цвет бордюра равный 7 (белый). Следовательно, первое значение будет равно 16+7=23, а второе - 0+7=7.Вообще то, Вы можете устанавливать любой цвет бордюра. Более того, можно создавать на нем различные цветовые эффекты. Для этого достаточно при инвертировании звукового бита использовать разные цвета бордюра.

Вы, наверное, заметили, что в приведенной выше программе на ассемблере, запрещаются прерывания. Думаю, стоит это пояснить. Дело в том, что ZX Spectrum устроен так, что каждую пятидесятую долю секунды вызывается подпрограмма ПЗУ, расположенная с адреса 56 (#38). И, если Вы хотите получить качественный (не трещащий) звуковой сигнал, то её надо отключить, что и достигается запрещением прерываний (команда DI). При возврате в бейсик прерывания необходимо снова разрешать (команда EI).

4. Программирование звука в кодах.

Простейший пример воспроизведения звукового сигнала в кодах был приведён в предыдущей главе. В принципе, это единственный способ получения звука, но оформить его можно по-разному. Например, как универсальную подпрограмму:

```
10          DI                 ; запрет прерываний
20 BEEP     LD    A,B          ; A=цвет бордюра
30          SET   4,A          ; бит D4=1
40          OUT   (254),A      ; вывод A в порт 254
50          PUSH  HL           ; сохранение HL
60 LOOP1    DEC   HL           ; HL=HL-1
70          LD    A,H          ; HL=
80          OR    L            ; 0 ?
90          JR    NZ,LOOP1     ; если нет, то цикл
100         POP   HL           ; восстановление HL
110         LD    A,C          ; A=цвет эффекта
120         OUT   (254),A      ; вывод A в порт 254
130         PUSH  HL           ; сохранение HL
140 LOOP2   DEC   HL           ; HL=HL-1
150         LD    A,H          ; HL=
160         OR    L            ; 0 ?
170         JR    NZ,LOOP2     ; если нет, то цикл
180         POP   HL           ; восстановление HL
190         DEC   DE           ; DE=DE-1
200         LD    A,D          ; DE=
210         OR    E            ; 0 ?
220         JR    NZ,BEEP      ; если нет, то цикл
230         EI                 ; разрешение прерываний
240         RET                ; возврат в бейсик
```

Перед вызовом этой подпрограммы необходимо занести в регистровую пару HL частоту, в DE - длительность, в B - цвет бордюра, а в C - цвет эффекта (если Вы хотите видеть бордюр однотонным, то значение C должно быть равно B). Кроме того, если к регистру B прибавить 8 (установить бит D3), то вместе с динамиком сигнал будет подаваться на магнитофон. Можно сделать эту программу универсальной несколько в другом смысле:

```
10          DI                 ; запрет прерываний
20          LD    A,(23624)    ; A=
30          SRL   A            ;   цвет
40          SRL   A            ;   бор-
50          SRL   A            ;   дюра
60 BEEP     XOR   16           ; инвертирование бита D4
70          OUT   (254),A      ; вывод A в порт 254
80          LD    C,A          ; сохранение A
90          PUSH  HL           ; сохранение HL
100 PAUSE   DEC   HL           ; HL=HL-1
110         LD    A,H          ; HL=
120         OR    L            ; 0 ?
130         JR    NZ,PAUSE     ; если нет, то цикл
140         POP   HL           ; восстановление HL
150         DEC   DE           ; DE=DE-1
160         LD    A,D          ; DE=
170         OR    E            ; 0 ?
180         LD    A,C          ; восстановление A
190         JR    NZ,BEEP      ; если DE<>0, то цикл
200         EI                 ; разрешение прерываний
210         RET                ; возврат в бейсик
```

Этой подпрограмме надо передать только частоту в регистре HL и длительность в DE, а цвет бордюра сохраняется тот, который был при её вызове (если только он не был установлен оператором OUT).

Примерно такая же подпрограмма имеется в ПЗУ ZX-Spectrum. Она расположена с адреса 949 (#03B5). Параметры,как и раньше, передаются в регистрах HL и DE. Их значения можно рассчитать по следующим формулам:

HL=INT(437500/f-30.125+.5)   DE=INT(f*t+.5)

- где f - частота в Гц, at- время в сек.

Частоту нот для пятой октавы можно взять из таблицы 2. Чтобы ноту повысить или понизить на октаву, её частоту надо соответственно умножить или разделить на 2. Причем, при делении значения получаются немного точнее, чем при умножении.

Нота	Обозначение	Частота, Гц	Октава
ДО	C	4186.01	5
ДО-диез	C#	4434.92	
РЕ	D	4698.64	
РЕ-диез	D#	4978.03	
МИ	E	5274.04	
ФА	F	5587.65	
ФА-диез	F#	5919.91	
СОЛЬ	G	6271.93	
СОЛЬ-диез	G#	6644.87	
ЛЯ	A	7040.00	
ЛЯ-диез	A#	7458.62	
СИ	B	7902.13	
ДО	C	8372.02	6

Табл. 2. Частоты нот для 5-ой октавы.

При генерации звука всеми способами, приведенными выше, необходимо учитывать, что чем выше тон, тем короче будет длительность.

Если Вы хотите при программировании звука в кодах использовать привычные параметры оператора BEEP, то Вы можете воспользоваться подпрограммой ПЗУ, расположенной по адресу 1016 (#03F8). Правда, с ней связаны некоторые трудности.

Дело в том, что параметры ей передаются через стек калькулятора, а засунуть туда дробное или отрицательное число не так-то просто.

Чтобы обойти это неудобство существует три основных способа. Первый - хранить необходимые значения в стандартной пятибайтовой форме и помещать их в стек с помощью специальной подпрограммы ПЗУ. Второй - хранить эти значения в символьной форме и помещать их в стек с помощью другой специальной подпрограммы ПЗУ. И третий - немножко изменить систему параметров и заняться вычислениями.

Первый способ не годится из-за слишком большого объема расходуемой памяти. Второй - по той же причине, плюс слишком медленная его работа. Остается третий способ. Он вполне подходит и его можно легко осуществить.

Пусть длительности задаются в сотых долях секунды, а высоты нот, как раньше, в полутонах выше или ниже ДО первой октавы. Такая система параметров позволяет получать ноты длительностью до 2.55 секунд с частотами в диапазоне от -60 до 69 (как в бейсике). Длительность ноты будет задаваться в регистре C, а высота - в регистре B.

Вот подпрограмма, осуществляющая все вышеописанное:

```
10          PUSH  BC          ; сохранение BC
20          LD    A,C         ; A=C (длительность)
30          CALL  11560       ; поместить A в стек калькулятора
40          LD    A,100       ; A=100
50          CALL  11560       ; поместить A в стек калькулятора
60          RST   40          ; вызов калькулятора
70          DEFB  5,56        ; деление длительности на 100
80          POP   BC          ; восстановление BC
90          LD    A,B         ; A=B (частота)
100         BIT   7,A         ; A - отрицательное ?
110         JR    NZ,MINUS    ; если да, то перейти на MINUS
120         CALL  11560       ; иначе, поместить A в стек калькулятора
130         JP    1016        ; вызов подпрограммы воспроизведения
140 MINUS   NEG               ; в A - абсолютное значение
150         CALL  11560       ; поместить A в стек калькулятора
160         RST   40          ; вызов калькулятора
170         DEFB  27,56       ; изменение знака
180         JP    1016        ; вызов подпрограммы воспроизведения
```

Если Ваш ассемблер не "переваривает" отрицательные числа (бывают такие "штучки"), то необходимые значения можно рассчитать по следующей формуле: n=256-x, - где x - абсолютная величина значения, а n- результат.

4.1. Программирование звуковых эффектов.

Вообще-то, программирование эффектов в кодах мало отличается от аналогичного занятия на бейсике, но Вы получаете значительное преимущество за счёт быстродействия. Поэтому, кроме чистого тона на ассемблере можно создавать и шум.

4.1.1. Тон.

Начнем с эффектов, основанных на генерации тона. Что они из себя представляют? Примерно то же, что и эффекты на бейсике: звук с плавно изменяющейся частотой. Для полного сходства можно даже использовать ту же подпрограмму из ПЗУ:

```
10          LD    B,30        ; B=количество нот
20          LD    HL,100      ; HL=начальная частота
30 LOOP     LD    DE,2        ; DE=длительность
40          PUSH  HL          ; сохранение HL
50          PUSH  BC          ; сохранение BC
60          CALL  949         ; вызов подпрограммы ПЗУ
70          POP   BC          ; восстановление BC
80          POP   HL          ; восстановление HL
90          LD    DE,25       ; DE=шаг изменения частоты
100         ADD   HL,DE       ; увеличение HL
110         DJNZ  LOOP        ; цикл
120         RET               ; возврат в бейсик
```

Скорее всего, Вы заметили, что в этом эффекте не запрещаются прерывания, а качество сигнала ничуть не ухудшилось (см. главу "Как получается звук"). Дело в том, что вызываемая подпрограмма все необходимое делает сама.

Если Вы уже успели набрать и запустить этот фрагмент, то, наверное, ощутили разницу в звучании, причем не в пользу бейсика.

Несколько модифицировать данный эффект можно изменяя не всю частоту, а только её младший байт:

```
10          LD    B,60        ; B=количество нот
20          LD    C,50        ; C=шаг изменения частоты
30          LD    HL,300      ; HL=начальная частота
40 LOOP     LD    DE,10       ; DE=длительность
50          PUSH  HL          ; сохранение HL
60          PUSH  BC          ; сохранение BC
70          CALL  949         ; вызов подпрограммы ПЗУ
80          POP   BC          ; восстановление BC
90          POP   HL          ; восстановление HL
100         LD    A,L         ; увели-
110         ADD   C           ;   чение
120         LD    L,A         ; L
130         DJNZ  LOOP        ; цикл
140         RET               ; возврат в бейсик
```

От использования подпрограмм ПЗУ перейдем к созданию собственных. Звучание предыдущих примеров можно сделать более плавным и протяжным:

```
10          DI                ; запрет прерываний
20          XOR   A           ; A=цвет бордюра (0)
30          LD    B,255       ; B=начальная частота
40          LD    C,255       ; C=длительность
50 BEEP     XOR   16          ; инвертирование бита D4
60          OUT   (254),A     ; вывод A в порт 254
70          PUSH  BC          ; сохранение BC
80 LOOP     DJNZ  LOOP        ; задержка
90          POP   BC          ; восстановление BC
100         DEC   B           ; уменьшение B
110         DEC   C           ; C=C-1
120         JR    NZ,BEEP     ; если C<>0, то цикл
```

```
130         EI                ; разрешение прерываний
140         RET               ; возврат в бейсик
```

Возможности этого эффекта легко увеличиваются:

```
10          DI                ; запрет прерываний
20          XOR   A           ; A=цвет бордюра (0)
30          LD    B,255       ; B=количество нот
40          LD    C,1         ; C=начальная частота
50 LOOP1    PUSH  BC          ; сохранение BC
60          LD    B,5         ; B=длительность
70 LOOP2    XOR   16          ; инвертирование бита D4
80          OUT   (254),A     ; вывод A в порт 254
90          PUSH  BC          ; сохранение BC
100         LD    B,C         ; B=C
110 LOOP3   DJNZ  LOOP3       ; задержка
120         POP   BC          ; восстановление BC
130         DJNZ  LOOP2       ; цикл
140         POP   BC          ; восстановление BC
150         INC   C           ; увеличение C
160         DJNZ  LOOP1       ; второй цикл
170         EI                ; разрешение прерываний
180         RET               ; возврат в бейсик
```

Скорее всего, Вы заметили, что в первом варианте задержка уменьшается, а во втором увеличивается. Ничто не мешает Вам сменить направление её изменения. Для этого только стоит выбрать команду DEC (уменьшение) или INC (увеличение). Также можно настроить и длительность, начальную частоту и т. п.

Можно изменить шаг смещения частоты:

```
10          DI                ; запрет прерываний
20          XOR   A           ; A=цвет бордюра (0)
30          LD    B,255       ; B=начальная частота
40          LD    C,255       ; C=длительность
50 BEEP     XOR   16          ; инвертирование бита D4
60          OUT   (254),A     ; вывод A в порт 254
70          PUSH  BC          ; сохранение BC
80 LOOP     DJNZ  LOOP        ; задержка
90          POP   BC          ; восстановление BC
100         EX    AF,AF'      ; смена регистров A и F на альтернативные
110         LD    A,B         ; A=B
120         SUB   3           ; A=A-3
130         LD    B,A         ; B=A
140         EX    AF,AF'      ; обратная смена регистров
150         DEC   C           ; C=C-1
160         JR    NZ,BEEP     ; если C<>0, то цикл
170         EI                ; разрешение прерываний
180         RET               ; возврат в бейсик
```

Теперь воспользуемся свободой действий, которую нам предоставляет программирование в кодах. Напишем нестандартную процедуру воспроизведения:

```
10          DI                ; запрет прерываний
20          LD    D,10        ; D=задержка 1
30          LD    E,100       ; E=задержка 2
40          LD    C,255       ; C=длительность
50          XOR   A           ; A=цвет бордюра (0)
60 LOOP1    XOR   16          ; инвертирование бита D4
70          OUT   (254),A     ; вывод A в порт 254
80          LD    B,D         ; B=D
90 LOOP2    DJNZ  LOOP2       ; задержка
100         XOR   16          ; инвертирование бита D4
110         OUT   (254),A     ; вывод A в порт 254
120         LD    B,E         ; B=E
130 LOOP3   DJNZ  LOOP3       ; задержка
140         INC   D           ; увеличение D
150         INC   E           ; увеличение E
160         DEC   C           ; C=C-1
170         JR    NZ,LOOP1    ; если C<>0, то цикл
```

```
180          EI                ; разрешение прерываний
190          RET               ; возврат в бейсик
```

Здесь тоже можно долго и старательно варьировать параметры (см. Приложение 1 - Flowing 2).

Довольно интересный эффект получается при использовании регистра R, значение младших семи битов которого увеличивается после выполнения очередного машинного цикла. Этот эффект можно условно назвать "полушумом".

Вот пример такого эффекта:

```
10           DI                ; запрет прерываний
20           LD    C,53        ; C=задержка 1
30           LD    B,207       ; B=задержка 2
40           LD    E,203       ; E=длительность
50           LD    D,0         ; D=цвет бордюра
60           LD    A,128       ; A=темп (0/128)
70           LD    R,A         ; R=A
80 BEGIN     LD    A,R         ; A=R
90 PAUS1     DEC   A           ; A=A-1
100          JR    NZ,PAUS1    ; если A<>0, то цикл
110          LD    A,D         ; A=D
120          OR    16          ; установка бита D4
130          OUT   (254),A     ; вывод A в порт 254
140          LD    A,C         ; A=C
150 PAUS2    DEC   A           ; A=A-1
160          JR    NZ,PAUS2    ; если A<>0, то цикл
170          LD    A,D         ; A=D
180          OUT   (254),A     ; вывод A в порт 254
190          LD    A,B         ; A=B
200 PAUS3    DEC   A           ; A=A-1
210          JR    NZ,PAUS3    ; если A<>0, то цикл
220          INC   C           ; C=C+1
230          INC   B           ; B=B+1
240          DEC   E           ; E=E-1
250          JR    NZ,BEGIN    ; если E<>0, то цикл
260          EI                ; разрешение прерываний
270          RET               ; возврат
```

Значения в строках 20 и 30 определяют задержки между перепадами уровня, в строке 40 - длительность эффекта, в строке 50 - цвет бордюра, а в строке 60 -темп. В строке 60 имеет смысл употреблять только значения 0 и 128, так как все остальные будут аналогичны этим. В строках 220 и 230 Вы можете установить закон изменения обеих задержек (команды INC, DEC и NOP с регистрами B и C в любой комбинации).

Если Вы не запутались во всех этих примерах, то перейдем к генерации шума. Если же Вы все-таки ничего не поняли, то советую испытать эти эффекты и поизменять их параметры.

4.1.2. Шум.

Что из себя представляет шум? Это последовательность импульсов случайной длительности. Поэтому для его создания нам понадобятся случайные данные. Где их взять? Единственный подходящий источник - ПЗУ, в котором записан интерпретатор бейсика. ПЗУ расположено с адреса 0 до 16383 (#3FFF). Правда, значения эти будут не совсем случайные, точнее совсем не случайные, но нас они устроят.

Создавать шум можно двумя разными способами. Они немного отличаются по звучанию. Первый из них состоит в выводе в порт значений, прочитанных из ПЗУ. Второй же - в использовании этих значений в качестве задержки.

При генерации шума любым из этих способов прерывания запрещать не обязательно, так как в шуме потрескивание слышно не будет.

При использовании первого способа из каждого байта можно извлечь данные на восемь проходов цикла воспроизведения, но это не выгодно с программной точки зрения. Поэтому, в большинстве эффектов из байта берется одно значение. Например:

```
10           LD    HL,0        ; HL=адрес ПЗУ
20           LD    BC,1000     ; BC=длительность
```

```
30 BEGIN    PUSH BC         ; сохранение BC
40          LD   A,(HL)     ; A=содержимое ячейки ПЗУ
50          AND  240        ; сброс битов бордюра
60          OUT  (254),A    ; вывод A в порт 254
70          LD   B,50       ; B=частота
80 LOOP     DJNZ LOOP       ; задержка
90          INC  HL         ; HL=HL+1
100         POP  BC         ; восстановление BC
110         DEC  BC         ; BC=BC-1
120         LD   A,B        ; BC=
130         OR   C          ; 0 ?
140         JR   NZ,BEGIN   ; цикл
150         RET             ; возврат в бейсик
```

Пример применения второго способа:

```
10          LD   HL,0       ; HL=адрес ПЗУ
20          LD   BC,1000    ; BC=длительность
30          XOR  A          ; A=цвет бордюра (0)
40 BEGIN    PUSH BC         ; сохранение BC
50          XOR  16         ; инвертирование бита D4
60          OUT  (254),A    ; вывод A в порт 254
70          LD   B,(HL)     ; B=содержимое ячейки ПЗУ
80 LOOP1    DJNZ LOOP1      ; задержка
90          LD   B,40       ; B=частота
100 LOOP2   DJNZ LOOP2      ; задержка
110         INC  HL         ; HL=HL+1
120         POP  BC         ; восстановление BC
130         DEC  BC         ; BC=BC-1
140         LD   D,A        ; сохранение A
150         LD   A,B        ; BC=
160         OR   C          ; 0 ?
170         LD   A,D        ; восстановление A
180         JR   NZ,BEGIN   ; цикл
190         RET             ; возврат в бейсик
```

Если в этих эффектах команду INC HL заменить на INC L, то их звучание станет как бы скачущим. Это происходит из-за того, что данные берутся не из всего заданного объема ПЗУ, а из его части размером 256 байт. Причем, когда эта часть кончается, чтение продолжается с её начала.

Попробуйте поизменять другие параметры.

Следующий шаг - шум с изменяющейся частотой. Выглядеть это будет примерно так:

```
10          LD   HL,0       ; HL=адрес ПЗУ
20          LD   B,100      ; B=длина эффекта
30          LD   C,10       ; C=начальная частота
40 LOOP1    PUSH BC         ; сохранение BC
50          LD   B,20       ; B=длительность
60 LOOP2    LD   A,(HL)     ; A=содержимое ячейки ПЗУ
70          AND  240        ; сброс битов бордюра
80          OUT  (254),A    ; вывод A в порт 254
90          PUSH BC         ; сохранение BC
100         LD   B,C        ; B=C
110 LOOP3   DJNZ LOOP3      ; задержка
120         INC  HL         ; HL=HL+1
130         POP  BC         ; восстановление BC
140         DJNZ LOOP2      ; цикл
150         POP  BC         ; восстановление BC
160         INC  C          ; увеличение задержки
170         DJNZ LOOP1      ; цикл
180         RET             ; возврат в бейсик
```

Этот эффект тоже можно изменить до неузнаваемости: сменить длину, частоту, длительность, шаг смещения частоты, направление смещения частоты (команда INC C или DEC C), вид шума (команда INC HL или INC L).

Еще один вариант шумового эффекта:

```
10          LD   HL,0       ; HL=адрес ПЗУ
20          LD   D,100      ; D=задержка 1
30          LD   E,10       ; E=задержка 2
```

```
40          LD    C,255       ; C=длительность
50          XOR   A           ; A=цвет бордюра (0)
60 BEGIN    XOR   16          ; инвертирование бита D4
70          OUT   (254),A     ; вывод A в порт 254
80          LD    B,(HL)      ; B=содержимое ячейки ПЗУ
90 LOOP1    DJNZ  LOOP1       ; задержка 1
100         LD    B,D         ; B=D
110 LOOP2   DJNZ  LOOP2       ; задержка 2
120         XOR   16          ; инвертирование бита D4
130         OUT   (254),A     ; вывод A в порт 254
140         LD    B,(HL)      ; B=содержимое ячейки ПЗУ
150 LOOP3   DJNZ  LOOP3       ; задержка 3
160         LD    B,E         ; B=E
170 LOOP4   DJNZ  LOOP4       ; задержка 4
180         INC   HL          ; HL=HL+1
190         INC   D           ; увеличение D
200         INC   E           ; увеличение E
210         DEC   C           ; C=C-1
220         JR    NZ,BEGIN    ; если C<>0, то цикл
230         RET               ; возврат в бейсик
```

Над этим примером можно издеваться так же долго, как и над всеми предыдущими.

Все эффекты, приведенные в данной главе, можно рассматривать как заготовки. Чтобы получить конечные варианты Вам, возможно, придется потрудиться. Как уже было сказано, все они имеют огромное число вариантов. Кроме того, Вы можете объединить несколько эффектов вместе, или вызывать их на выполнение в цикле и т. п. Все основные принципы комбинации эффектов перечислены в главе 2.1.

Все эффекты написаны так, чтобы можно было изменять максимальное число параметров. В случаях конкретного применения их можно значительно упрощать. Вот пример скомбинированного и упрощенного эффекта:

```
10          DI                ; запрет прерываний
20          LD    E,100       ; E=длительность цикла
30          LD    C,0         ; C=цвет бордюра
40          LD    B,4         ; B=число циклов
50          LD    L,1         ; L=смещение частоты
60          LD    H,30        ; H=начальная частота
70 LOOP1    LD    D,E         ; D=E
80 LOOP2    LD    A,C         ; A=C
90          XOR   16          ; инвертирование бита D4
100         OUT   (254),A     ; вывод A в порт 254
110         LD    C,A         ; C=A
120         LD    A,H         ; A=H
130         ADD   A,L         ; прибавить L к A
140         LD    H,A         ; H=A
150 LOOP3   DEC   A           ; A=A-1
160         JR    NZ,LOOP3    ; если A<>0, то цикл
170         DEC   D           ; D=D-1
180         JR    NZ,LOOP2    ; если D<>0, то цикл
190         LD    A,L         ; A=L
200         NEG               ; изменение знака регистра A
210         LD    L,A         ; L=A
220         DJNZ  LOOP1       ; цикл
230         EI                ; разрешение прерываний
240         RET               ; возврат
```

Все вышеприведенные эффекты Вы можете настроить под собственные нужды. Например, изменить цвет бордюра или сделать так, чтобы сигнал кроме динамика выводился на магнитофон (для этого все команды XOR 16 надо заменить на XOR 24, а AND 240 на AND 248).

4.1.3. Комплексы эффектов.

Обычно в играх (да и в любых других программах) используется не один и не два звуковых эффекта, а гораздо больше. Поэтому эффекты удобно объединять в группы (комплексы) и вызывать их на исполнение одной подпрограммой, передавая ей в качестве параметра номер

эффекта.

Если эффекты разнообразны и воспроизводятся различными подпрограммами, то в таблице эффектов лучше всего хранить адреса этих подпрограмм. В таком случае для воспроизведения эффектов можно воспользоваться такой подпрограммой:

```
10          ADD   A,A         ; A=A*2
20          LD    E,A         ; DE
30          LD    D,0         ; =A
40          LD    HL,TABLE    ; HL=адрес таблицы
50          ADD   HL,DE       ; HL=HL+DE
60          LD    E,(HL)      ; DE=
70          INC   HL          ; адрес
80          LD    D,(HL)      ; подпрограммы
90          EX    DE,HL       ; обменять значениями HL и DE
100         JP    (HL)        ; вызов подпрограммы эффекта
110 TABLE   DEFW  …           ; таблица адресов
```

Перед вызовом этой подпрограммы в регистр A надо занести номер эффекта. Не забудьте также заполнить таблицу адресами эффектов. Естественно, сами эффекты должны располагаться по указанным адресам. Ни в коем случае не стоит указывать номер эффекта больше числа описанных в таблице подпрограмм, иначе компьютер "зависнет" или "сбросится". Учтите также, что эта подпрограмма может работать максимум со 127 эффектами.

Если эффекты однотипные и воспроизводятся одной и той же подпрограммой, то в таблице эффектов можно хранить параметры этой подпрограммы. Вот пример, использующий данный способ:

```
10          LD    E,A         ; A
20          ADD   A,A         ;  =
30          ADD   A,E         ;   A*3
40          LD    E,A         ; DE
50          LD    D,0         ;   =A
60          LD    HL,TABLE    ; HL=адрес таблицы
70          ADD   HL,DE       ; HL=HL+DE
80          LD    C,(HL)      ; C=длительность
90          INC   HL          ; HL=HL+1
100         LD    E,(HL)      ; E=частота
110         INC   HL          ; HL=HL+1
120         LD    A,(HL)      ; A=изменение частоты
130         LD    (CHNG),A    ; установка изменения частоты
140         DI                ; запрет прерываний
150         XOR   A           ; A=цвет бордюра (0)
160 BEGIN   XOR   16          ; инвертирование бита D4
170         OUT   (254),A     ; вывод A в порт 254
180         LD    B,E         ; B=E
190 PAUSE   DJNZ  PAUSE       ; задержка
200 CHNG    NOP               ; резерв для изменения частоты
210         DEC   C           ; C=C-1
220         JR    NZ,BEGIN    ; если C<>0, то цикл
230         EI                ; разрешение прерываний
240         RET               ; возврат
250 TABLE   DEFB  0,0,28      ; таблица
260         DEFB  0,0,29      ;
270         DEFB  0,128,28    ;   эффектов
280         DEFB  0,128,29    ;
```

Перед вызовом этой подпрограммы в регистр A надо занести номер эффекта. В приведенной программе уже созданы четыре эффекта, но Вы можете изменить их или увеличить их число. При вызове эффекта, номер которого больше числа описанных подпрограмм, также может произойти что-нибудь ужасное.

В этом примере на описание эффекта отводится три байта. Первый байт - длительность эффекта, второй - начальная частота, а третий - способ изменения частоты. Третий байт может принимать значения 0,28 и 29. Что означает, соответственно, сохранение, увеличение и уменьшение частоты. Заносить какие-либо другие значения в этот байт не стоит, так как это может привести к непредсказуемым последствиям.

Приведенная программа является только примером. Вы можете настроить её для работы с абсолютно любым эффектом.

Эта подпрограмма может работать максимум с 85 эффектами.

В обеих приведенных подпрограммах нумерация эффектов начинается с нуля.

4.2. Управление громкостью.

Вообще-то, ZX-Spectrum абсолютно не приспособлен для управления громкостью, но её все-таки можно изменять в небольших пределах благодаря относительно высокой тактовой частоте микропроцессора Z-80 (около 3.5 МГц).

Как уже говорилось в главе 2, когда Вы посылаете сигнал на динамик, его мембрана изменяет свое положение. Если во время её движения послать обратный сигнал, то она вернется в исходное положение, не пройдя нужного пути. Таким образом, амплитуда (соответственно и громкость) будет меньше обычной. Для примера, сравните звучание таких двух практически одинаковых программ:

```
10 PROG1     DI                 ; запрет прерываний
20           LD    B,255        ; B=частота
30           LD    C,255        ; C=длительность
40           XOR   A            ; A=цвет бордюра (0)
50 LOOP1     XOR   16           ; инвертирование бита D4
60           OUT   (254),A      ; вывод A в порт 254
70           PUSH  BC           ; сохранение BC
80 LOOP2     DJNZ  LOOP2        ; задержка
90           POP   BC           ; восстановление BC
100          DEC   C            ; C=C-1
110          JR    NZ,LOOP1     ; если C<>0, то цикл
120          EI                 ; разрешение прерываний
130          RET                ; возврат в бейсик

10 PROG2     DI                 ;
20           LD    B,255        ;
30           LD    C,255        ;
40           XOR   A            ;
50 LOOP1     XOR   16           ; все
60           OUT   (254),A      ;
70           XOR   16           ; (кроме этих двух строк,
80           OUT   (254),A      ; повторяющих предыдущие)
90           PUSH  BC           ;
100 LOOP2    DJNZ  LOOP2        ; аналогично
110          POP   BC           ;
120          DEC   C            ; предыдущему
130          JR    NZ,LOOP1     ;
140          EI                 ; примеру !
150          RET                ;
```

Когда Вы их наберете на ассемблере и запустите, Вы сильно удивитесь. Однако против фактов не попрёшь - вторая подпрограмма будет звучать в несколько раз тише первой!

Приведу текст более универсальной подпрограммы:

```
10           DI                 ; запрет прерываний
20           LD    B,50         ; B=громкость
30           LD    C,50         ; C=частота
40           LD    D,15         ; D=длительность
50           LD    E,50         ; E=количество нот
60           XOR   A            ; A=цвет бордюра (0)
70 LOOP1     PUSH  DE           ; сохранение DE
80 LOOP2     PUSH  BC           ; сохранение BC
90           XOR   16           ; инвертирование бита D4
100          OUT   (254),A      ; вывод A в порт 254
110 LOOP3    DJNZ  LOOP3        ; задержка (громкость)
120          XOR   16           ; инвертирование бита D4
130          OUT   (254),A      ; вывод A в порт 254
140          LD    B,C          ; B=C
150 LOOP4    DJNZ  LOOP4        ; задержка (частота)
```

```
160           POP   BC           ; восстановление BC
170           DEC   D            ; D=D-1
180           JR    NZ,LOOP2     ; если D<>0, то цикл
190           POP   DE           ; восстановление DE
200           NOP                ; резерв
210           NOP                ; резерв
220           NOP                ; резерв
230           DEC   E            ; E=E-1
240           JR    NZ,LOOP1     ; если E<>0, то цикл
250           EI                 ; разрешение прерываний
260           RET                ; возврат в бейсик
```

В этом эффекте Вы можете подобрать нужную громкость. Но не очень увлекайтесь: при значениях больших, чем 40 - 50 громкость уже не увеличивается. Необходимо заметить, что при изменении громкости во время воспроизведения меняется и частота. Поэтому надо предусмотреть её компенсацию.

Теперь о трех байтах резерва (команды NOP). Они предназначены для изменения громкости (регистр B), частоты (регистр C) и длительности (регистр D). Если Вы замените эти три NOPа на команды DEC B (уменьшение громкости), INC C (компенсация частоты) и INC D (увеличение длительности), то у Вас получится интересный эффект с затухающей громкостью.

4.3. Управление тембром.

Для управления тембром ZX-Spectrum так же не приспособлен, как и для управления громкостью. Но, как и громкостью, им можно управлять в небольших пределах.

Как Вы уже знаете, тембр зависит от формы звуковой волны. Так как приспособить для формирования тембра подпрограмму управления громкостью не так то просто, при создании различных форм волны придется обходиться только двумя уровнями громкости. Сделать это можно, например, с помощью такой подпрограммы:

```
10            DI                 ; запрет прерываний
20 LOOP1      LD    B,8          ; B=темп
30 LOOP2      XOR   A            ; A=цвет бордюра (0)
40            RLC   E            ; скроллинг тембра через флаг CY
50            JR    NC,NOSIGN    ; если флаг CY=0, то перейти на NOSIGN
60            OR    16           ; установка бита D4 регистра A
70 NOSIGN     OUT   (254),A      ; вывод A в порт 254
80            LD    D,H          ; D=задержка
90 PAUSE      DEC   D            ; D=D-1
100           JR    NZ,PAUSE     ; если D<>0, то цикл
110           DJNZ  LOOP2        ; цикл темпа
120           LD    A,H          ; A=H
130           ADD   A,L          ; прибавить L к A
140           LD    H,A          ; H=A
150           DEC   C            ; C=C-1
160           JR    NZ,LOOP1     ; если C<>0, то цикл
170           EI                 ; разрешение прерываний
180           RET                ; возврат
```

Перед вызовом этой подпрограммы необходимо в регистр C занести длительность ноты, в регистр H - частоту, в регистр L - смещение частоты (возможны отрицательные значения и ноль), а в регистр E - тембр. Учтите, что не все значения регистра E имеют смысл. Так, при значении E равном 255 или 0 Вы вообще ничего не услышите, а тембры 1 и 128 будут звучать абсолютно одинаково и т. д.

По существу, имеет смысл использовать только следующие 19 тембров: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 37, 43, 45, 51 и 85. Все остальные значения на слух воспринимаются аналогично этим.

Приведенная программа является только примером. Вы вполне можете написать свою, воспроизводящую звук с 16-ти битовым тембром, или с тембром, заранее записанным в память с магнитофонного входа (см. главу 4.6).

4.4. Программирование музыки.

Программирование музыки в кодах - занятие неблагодарное. Лучше это делать с помощью какого-нибудь музыкального редактора. Но редактор может не удовлетворить все Ваши потребности, так что освоить программирование в кодах все же желательно.

Принцип тот же - последовательно читаются данные, и вызывается процедура воспроизведения. При этом необходимо предусмотреть прерывание исполнения (чаще всего в этих целях контролируется нажатие какой-нибудь клавиши). Например:

```
10 START    LD    HL,60000    ; HL=адрес данных
20 NEXT     LD    E,(HL)      ; DE=
30          INC   HL          ;   дли-
40          LD    D,(HL)      ;    тель
50          INC   HL          ;     ность
60          LD    A,D         ; DE
70          CP    255         ;   =
80          JR    NZ,CONT     ;    65535
90          LD    A,E         ;      ?
100         CP    255         ; если да,
110         JR    Z,START     ; то мелодия сначала
120 CONT    LD    A,D         ; DE=
130         OR    E           ;    0 ?
140         JR    NZ,BEEP     ; если нет, то перейти на BEEP
150         LD    E,(HL)      ; DE=
160         INC   HL          ;    длитель-
170         LD    D,(HL)      ;    ность
180         INC   HL          ;    паузы
190 PAUSE   HALT              ; ожидание прерывания
200         DEC   DE          ; DE=DE-1
210         LD    A,D         ; DE=
220         OR    E           ;    0 ?
230         JR    NZ,PAUSE    ; если нет, то цикл
240         JR    NEXT        ; переход на обработку след. данных
250 BEEP    LD    C,(HL)      ; BC=
260         INC   HL          ;    час-
270         LD    B,(HL)      ;    то-
280         INC   HL          ;    та
290         PUSH  HL          ; сохранение HL
300         PUSH  BC          ; поместить BC в стек
310         POP   HL          ; поместить значение из стека в HL
320         CALL  949         ; вызов подпрограммы воспроизведения
330         CALL  654         ; контроль клавиатуры
340         POP   HL          ; восстановление HL
350         LD    A,E         ; A=E
360         CP    255         ; если нажата клавиша,
370         RET   NZ          ; то возврат в бейсик
380         JR    NEXT        ; переход на обработку след. данных
```

При вызове этой подпрограммы из машинных кодов обратите внимание, что прерывания должны быть разрешены! Иначе, если в данных первой будет стоять пауза, компьютер "зависнет".

Перед использованием этой довольно страшненькой подпрограммы необходимо создать для нее массив данных (это и есть самое веселое). В данном случае он должен располагаться с адреса 60000 (#EA60), но Вы можете легко изменить его местонахождение, поменяв число в первой строке этой подпрограммы.

Данные для каждой ноты должны быть следующими: сначала два байта длительности (первый - младший), затем два байта частоты. Если вместо длительности поместить два нуля, то следующая пара байт будет рассматриваться как длина паузы. Чтобы отметить конец мелодии, вставьте в данные вместо очередной длительности два байта 255.

Длительность паузы измеряется в пятидесятых долях секунды, а в чем измеряется длительность ноты не знает, наверное, никто, хотя рассчитать необходимое значение все же можно (см. главу 4).

Теперь о недостатках, которых, честно говоря, хоть отбавляй. Два из них уже были упомянуты выше: трудность составления мелодии и довольно странное измерение длительности

нот. Кроме того, длительность зависит от частоты. Например, если Вы введете такие данные: 0,1,100,0,0,1,0,5,255,255, то не услышите две разные ноты одинаковой длины, как ожидали.

Ещё один недостаток - возможность прервать исполнение только в промежутке между нотами. Впрочем, это можно легко исправить:

```
10 START    LD    HL,60000    ; HL=адрес данных
20 NEXT     LD    E,(HL)      ; DE=
30          INC   HL          ;    дли-
40          LD    D,(HL)      ;    тель
50          INC   HL          ;    ность
60          LD    A,D         ; DE
70          CP    255         ;   =
80          JR    NZ,CONT     ;    65535
90          LD    A,E         ;      ?
100         CP    255         ; если да,
110         JR    Z,START     ; то мелодия сначала
120 CONT    LD    A,D         ; DE=
130         OR    E           ;    0 ?
140         JR    NZ,BEEP     ; если нет, то перейти на BEEP
150         LD    E,(HL)      ; DE=
160         INC   HL          ;    длитель-
170         LD    D,(HL)      ;    ность
180         INC   HL          ;    паузы
190 PAUSE   XOR   A           ; A=0 (контроль всей клавиатуры)
200         IN    A,(254)     ; опрос клавиш
210         CPL               ; инвертирование A
220         AND   31          ; сброс лишних битов
230         RET   NZ          ; если нажата клавиша, то возврат
240         HALT              ; ожидание прерывания
250         DEC   DE          ; DE=DE-1
260         LD    A,D         ; DE=
270         OR    E           ;    0 ?
280         JR    NZ,PAUSE    ; если нет, то цикл
290         JR    NEXT        ; переход на обработку след. данных
300 BEEP    LD    C,(HL)      ; BC=
310         INC   HL          ;    час-
320         LD    B,(HL)      ;    то-
330         INC   HL          ;    та
340         PUSH  HL          ; сохранение HL
350         PUSH  BC          ; поместить BC в стек
360         POP   HL          ; поместить значение из стека в HL
370         CALL  PLAY        ; вызов подпрограммы воспроизведения
380         POP   HL          ; восстановление HL
390         RET   NZ          ; то возврат в бейсик
400         JR    NEXT        ; переход на обработку след. данных
410 PLAY    DI                ; запрет прерываний
420         LD    A,(23624)   ; A=
430         SRL   A           ;   цвет
440         SRL   A           ;   бор-
450         SRL   A           ;   дюра
460 LOOP1   XOR   16          ; инвертирование бита D4
470         OUT   (254),A     ; вывод A в порт 254
480         LD    C,A         ; сохранение A
490         PUSH  HL          ; сохранение HL
500 LOOP2   XOR   A           ; A=0 (контроль всей клавиатуры)
510         IN    A,(254)     ; опрос клавиш
520         CPL               ; инвертирование A
530         AND   31          ; сброс лишних битов
540         JR    Z,CONT2     ; если клавиша не нажата, то продолжить
550         POP   HL          ; снять значение со стека
560         EI                ; разрешение прерываний
570         RET               ; возврат
580 CONT2   DEC   HL          ; HL=HL-1
590         LD    A,H         ; HL=
600         OR    L           ;    0 ?
```

```
610          JR    NZ,LOOP2    ; если нет, то цикл
620          POP   HL          ; восстановление HL
630          DEC   DE          ; DE=DE-1
640          LD    A,D         ; DE=
650          OR    E           ;    0 ?
660          LD    A,C         ; восстановление A
670          JR    NZ,LOOP1    ; если DE<>0, то цикл
680          EI                ; разрешение прерываний
690          RET               ; возврат
```

Обратите внимание на то, что одинаковые данные будут воспроизводиться по-разному в этой и предыдущей подпрограммами.

И, наконец, самый очевидный недостаток - музыка, которая получается с помощью этих подпрограмм слишком примитивна. На ассемблере можно сделать что-нибудь и посерьёзнее.

Надеюсь, что убедил Вас не пытаться программировать музыку напрямую в кодах, а вместо этого приобрести хороший музыкальный редактор.

Все подпрограммы, приведенные в этой главе, можно использовать и для создания эффектов. Для этого достаточно вместо нот закодировать отдельные ступени эффекта. Желательно, чтобы каждая ступень была как можно короче, тогда эффект будет звучать плавно.

4.5. Многоголосые мелодии.

Хотя в предыдущей главе я старательно доказывал Вам тщетность попыток программирования музыки на ассемблере, Вы все-таки должны знать основные принципы этого занятия.

Сейчас я опишу создание многоголосой музыки, то есть музыки, в которой может звучать две или больше ноты одновременно.

Первым примером будет подпрограмма, имитирующая двухголосое звучание:

```
10           DI                ; запрет прерываний
20           LD    D,200       ; D=частота 1
30           LD    E,50        ; E=частота 2
40           LD    H,150       ; H=длительность
50           XOR   A           ; A=бордюр для голоса 1 (0)
60           EX    AF,AF'      ; смена регистров A и F на альтернативные
70           XOR   A           ; A=бордюр для голоса 2 (0)
80           LD    C,E         ; C=счетчик 1
90           LD    B,D         ; B=счетчик 2
100 BEEP     EX    AF,AF'      ; смена регистров A и F (смена голоса)
110          DEC   C           ; C=C-1
120          JR    NZ,CONT     ; если C<>0, топерейти на CONT
130          LD    C,E         ; восстановить счетчик 1
140          XOR   16          ; инвертировать бит D4 голоса 1
150 CONT     OUT   (254),A     ; вывести A в порт 254
160          EX    AF,AF'      ; смена регистров A и F (смена голоса)
170          DEC   B           ; B=B-1
180          JR    NZ,CONT2    ; если B<>0, то перейти на CONT2
190          LD    B,D         ; восстановить счетчик 2
200          XOR   16          ; инвертировать бит D4 голоса 2
210 CONT2    OUT   (254),A     ; вывести A в порт 254
220          INC   L           ; L=L+1
230          JR    NZ,BEEP     ; если L<>0, то перейти на BEEP
240          NOP               ; резерв
250          NOP               ; резерв
260          DEC   H           ; H=H-1
270          JR    NZ,BEEP     ; если H<>0, то перейти на BEEP
280          EI                ; разрешение прерываний
290          RET               ; возврат
```

Теперь поясню некоторые детали.

Два альтернативных регистра A используются первым и вторым голосами для хранения цвета бордюра и состояния динамика.

Странная операция с регистром L перед завершением цикла предназначена для "растягивания" длительности. Если её убрать, то звучание этой подпрограммы станет настолько

кратковременным, что Вы можете его даже не услышать.

Два байта резерва (операции NOP) Вы можете использовать для изменения частоты обоих голосов во время воспроизведения. Для этого вместо "NOP" нужно вставить INC D или DEC D для первого голоса, и INC E или DEC E для второго.

Эту подпрограмму можно использовать на ассемблере (см. предыдущую главу) или на бейсике (в данном случае параметры следует заносить оператором POKE в следующие ячейки: ADDR+2 - частота 1, ADDR+4 - частота 2, ADDR+6 - длительность, где ADDR - адрес расположения подпрограммы в памяти).

Следующие две многоголосые подпрограммы очень похожи друг на друга, но первая звучит значительно тише и несколько чище второй.

Вот они:

```
10          DI                  ; запрет прерываний
20          LD    HL,2000       ; HL=длительность
30 LOOP1    LD    IX,FRQS       ; IX=адрес таблицы частот
40          LD    B,5           ; B=число голосов
50 LOOP2    DEC   (IX+0)        ; байт по адресу IX+0 уменьшить на 1
60          JR    NZ,NOSIGN     ; если не 0, то перейти на NOSIGN
70          LD    A,24          ; A=цвет бордюра + 24
80          OUT   (254),A       ; вывод A в порт 254
90          LD    A,(IX+5)      ; обновление счетчика
100         LD    (IX+0),A      ;   текущего канала
110         XOR   A             ; A=цвет бордюра (0)
120         OUT   (254),A       ; вывод A в порт 254
130 NOSIGN  INC   IX            ; IX=IX+1
140         DJNZ  LOOP2         ; цикл
150         DEC   HL            ; HL=HL-1
160         LD    A,H           ; HL=
170         OR    L             ;    0?
180         JR    NZ,LOOP1      ; если нет, то цикл
190         EI                  ; разрешение прерываний
200         RET                 ; возврат
210 FRQS    DEFB  1,1,1,1,1     ; счетчики для 5 каналов
220         DEFB  10,20,30      ; частоты для
230         DEFB  40,50         ;    5 каналов
```

```
10          DI                  ; запрет прерываний
20          LD    HL,2000       ; HL=длительность
30 LOOP1    LD    IX,FRQS       ; IX=адрес таблицы частот
40          LD    B,5           ; B=число голосов
50 LOOP2    DEC   (IX+0)        ; байт по адресу IX+0 уменьшить на 1
60          JR    NZ,NOSIGN     ; если не 0, то перейти на NOSIGN
70          LD    A,(IX+5)      ; A=содержимое IX+5
80          XOR   1             ; инвертирование бита D0
90          LD    (IX+5),A      ; содержимое IX+5=A
100         LD    A,0           ; A=цвет бордюра
110         JR    Z,NOSIGN      ; если D0 содер. IX+5=0,то перейти на NOSIGN
120         OR    24            ; установить в 1 биты D3 и D4 регистра A
130         OUT   (254),A       ; вывод A в порт 254
140         LD    A,(IX+10)     ; обновление счетчика
150         LD    (IX+0),A      ;   текущего канала
160 NOSIGN  INC   IX            ; IX=IX+1
170         DJNZ  LOOP2         ; цикл
180         DEC   HL            ; HL=HL-1
190         LD    A,H           ; HL=
200         OR    L             ;    0?
210         JR    NZ,LOOP1      ; если нет, то цикл
220         EI                  ; разрешение прерываний
230         RET                 ; возврат
240 FRQS    DEFB  1,1,1,1,1     ; счетчики для 5 каналов
250         DEFB  0,0,0,0,0     ; буферы для 5 каналов
260         DEFB  10,20,30      ; частоты для
270         DEFB  40,50         ;    5 каналов
```

Эти подпрограммы заводят счетчики для каждого из каналов и последовательно их уменьшают. Если какой-либо из счетчиков становится равным 0, на динамик выводится сигнал, а в данный счетчик заносится частота из таблицы.

В данных примерах подпрограммы воспроизводят по пять голосов, но Вы можете легко изменить их число, модифицировав значения в строках 40 и 90 для первой подпрограммы или 40, 70, 90 и 140 (в строку 140 надо вставить число голосов, умноженное на 2) - для второй. При этом надо также скорректировать таблицы частот.

Хотя голосов может быть довольно много (до 85), увлекаться их количеством не стоит, так как чем их больше, тем хуже качество. Впрочем, вряд ли найдется гений, способный написать мелодию на 85 голосов.

Если Вы хотите использовать многоголосие, но в конкретный момент времени Вам надо воспроизвести меньшее число голосов, чем задано, то неиспользуемым голосам надо присвоить частоты, совпадающие с частотами используемых голосов.

4.6. Обработка внешних сигналов.

ZX-Spectrum предоставляет программистам довольно интересную возможность - считывать и сохранять в памяти звуковые сигналы, подаваемые на его магнитофонный вход. Эти фрагменты можно воспроизводить с различной скоростью, что, несомненно, очень интересно. Правда, качество полученного таким образом звука оставляет желать лучшего, а его максимальная длительность ограничивается несколькими десятками секунд (просто-напросто кончаются 48К ОЗУ!).

Но перейдем от слов к делу. Вот подпрограмма, записывающая звук в память:

```
10          DI                  ; запрет прерываний
20          LD    HL,25000      ; HL=адрес сохранения звука
30          LD    DE,35000      ; DE=длительность звука
40 LOOP1    LD    B,8           ; B=счетчик битов
50 LOOP2    SLA   (HL)          ; скроллинг значения в памяти
60          IN    A,(254)       ; ввести значение из порта 254
70          BIT   6,A           ; проверить бит магнитофона
80          JR    Z,NOSIGN      ; если 0, то перейти на NOSIGN
90          SET   0,(HL)        ; установить бит D0 в памяти
100 NOSIGN  LD    C,3           ; C=задержка
110 PAUSE   DEC   C             ; C=C-1
120         JR    NZ,PAUSE      ; если C<>0, то цикл
130         DJNZ  LOOP2         ; продолжить цикл обработки байта
140         INC   HL            ; HL=HL+1
150         DEC   DE            ; DE=DE-1
160         LD    A,D           ; DE=
170         OR    E             ;    0?
180         JR    NZ,LOOP1      ; если нет, то цикл
190         EI                  ; разрешение прерываний
200         RET                 ; возврат
```

А это подпрограмма воспроизведения:

```
10          DI                  ; запрет прерываний
20          LD    HL,25000      ; HL=адрес сохраненного звука
30          LD    DE,35000      ; DE=длительность звука
40          NOP                 ; резерв
50          XOR   A             ; A=цвет бордюра (0)
60 LOOP3    LD    B,8           ; B=счетчик битов
70 LOOP4    AND   239           ; сброс бита D4 регистра A
80          RLC   (HL)          ; скроллинг данных через флаг CY
90          JR    NC,NOSGN      ; если флаг CY=0, то перейти на NOSGN
100         OR    16            ; установка бита D4 регистра A
110 NOSGN   OUT   (254),A       ; вывод A в порт 254
120         LD    C,3           ; C=задержка
130 PAUS2   DEC   C             ; C=C-1
140         JR    NZ,PAUS2      ; если C<>0, то цикл
150         DJNZ  LOOP4         ; продолжить цикл обработки байта
160         INC   HL            ; HL=HL+1
170         LD    C,A           ; сохранение A
```

```
180          DEC   DE          ; DE=DE-1
190          LD    A,D         ; DE=
200          OR    E           ;    0?
210          LD    A,C         ; восстановление A
220          JR    NZ,LOOP3    ; если DE<>0, то цикл
230          EI                ; разрешение прерываний
240          RET               ; возврат
```

Адреса данных и их длина должны совпадать в этих двух подпрограммах. Обратите также внимание на то, что они используют 35Кб памяти с адреса 25000. Поэтому в этой области ОЗУ не должно быть никаких данных. Сами подпрограммы тоже должны располагаться где-нибудь в другом месте. Оптимальным вариантом будет такой: перенумеруйте строки второй подпрограммы, начиная с 210 и с шагом 10 (если Вы работаете в GENS4 -введите для этих целей команду N210,10); соедините то, что получилось с первой подпрограммой и введите следующие строки:

```
7            ORG   24900
20           LD    HL,BUFFER
30           LD    DE,LENGTH
220          LD    HL,BUFFER
230          LD    DE,LENGTH
450 LENGTH   EQU   35000       ; LENGTH=35000
460 BUFFER   NOP               ; здесь располагается буфер
```

Проследите также, чтобы ассемблер располагался приблизительно с адреса 40000. В любом случае, он будет стёрт из памяти при вызове подпрограммы ввода звука. Поэтому, не стоит вызывать её из обреченного ассемблера.

Ещё одно важное замечание: при использовании этих подпрограмм следите, чтобы выражение BUFFER+LENGTH не было больше 65535, иначе часть звуковых данных просто не запишется, а вместо них Вы услышите шум.

В строке 100 первой подпрограммы и в строке 120 второй, в регистр C заносится задержка между выборками данных. Если эти два значения равны, то звук будет воспроизводиться с реальной скоростью. Если второе число меньше первого, то звук будет ускорен, а если больше - замедлён. Причем, чем меньше задержка в подпрограмме ввода, тем выше качество вводимого сигнала, но меньше его длительность.

Байт резерва в подпрограмме воспроизведения предназначен для переворачивания звука, то есть проигрывания его с конца на начало. Для включения этого эффекта надо заменить операцию NOP в этом байте на ADD HL,DE, а INC HL в строке 160 на DEC HL.

Существует и другой способ записи звука. Он основан на подсчете звуковых импульсов. Перед первым способом он не имеет никаких особых преимуществ, за исключением возможности незначительного понижения уровня шума. Вот подпрограмма, записывающая звук вторым способом:

```
10           DI                ; запрет прерываний
20           LD    HL,25000    ; HL=адрес сохранения звука
30           LD    DE,35000    ; DE=длительность звука
40           LD    BC,511      ; C - буфер, B - счетчик
50 LOOP1     IN    A,(254)     ; ввод значения из порта 254
60           EXX               ; смена набора регистров
70           LD    B,1         ; B=задержка
80 PAUSE     DJNZ  PAUSE       ; задержка
90           EXX               ; смена набора регистров
100          BIT   6,A         ; проверить бит магнитофона
110          LD    A,0         ; A=0
120          JR    NZ,SIGN     ; если бит D6=1, то перейти на SIGN
130          DEC   A           ; A=255
140 SIGN     CP    C           ; A=C (предыдущему значению)?
150          JR    NZ,WRITE    ; если нет, то перейти на WRITE
160          INC   B           ; B=B+1 (счетчик импульсов)
170          JR    NZ,LOOP1    ; если B<>0, то цикл
180 WRITE    DEC   B           ; B=B-1
190          JR    NZ,POKE     ; если B<>0, то перейти на POKE
200          LD    B,2         ; B=2
210          CPL               ; инвертирование A
```

```
220         LD    C,A         ; C=A
230         JR    LOOP1       ; перейти на LOOP1
240 POKE    INC   B           ; B=B+1
250         LD    (HL),B      ; запись значения в память
260         LD    B,1         ; B=1
270         LD    C,A         ; C=A
280         INC   HL          ; HL=HL+1
290         DEC   DE          ; DE=DE-1
300         LD    A,D         ; DE=
310         OR    E           ;    0?
320         JR    NZ,LOOP1    ; если нет, то цикл
330         EI                ; разрешение прерываний
340         RET               ; возврат
```

В строке 20 задается адрес сохранения звука, в строке 30 - длительность вводимого звука, в строке 70 - задержка между выборками сигнала.

Строки 180 - 240 обеспечивают шумопонижение.

В регистре C сохраняется значение предыдущей выборки, а регистр B служит счетчиком одинаковых выборок.

А вот подпрограмма воспроизведения:

```
10          DI                ; запрет прерываний
20          LD    HL,25000    ; HL=адрес сохраненного звука
30          LD    DE,35000    ; DE=длительность звука
40          NOP               ; резерв
50          XOR   A           ; A=цвет бордюра (0)
60 LOOP2    LD    B,(HL)      ; B=очередная длительность
70 LOOP3    OUT   (254),A     ; вывод A в порт 254
80          EXX               ; смена набора регистров
90          LD    B,1         ; B=задержка
100 PAUS2   DJNZ  PAUS2       ; задержка
110         EXX               ; смена набора регистров
120         CALL  124         ; компенсационная
130         BIT   0,(HL)      ;   задержка
140         DJNZ  LOOP3       ; цикл
150         LD    B,(HL)      ; B=текущая длительность
160         DEC   B           ; B=
170         INC   B           ;   0?
180         JR    Z,CONT      ; если да, то перейти на CONT
190         XOR   16          ; инвертирование бита D4
200 CONT    INC   HL          ; HL=HL+1
210         DEC   DE          ; DE=DE-1
220         LD    B,A         ; сохранение A
230         LD    A,D         ; DE=
240         OR    E           ;    0?
250         LD    A,B         ; восстановление A
260         JR    NZ,LOOP2    ; если DE<>0, то цикл
270         EI                ; разрешение прерываний
280         RET               ; возврат
```

В строке 20 задается адрес сохраненного звука, в строке 30 - его длительность, а в строке 90 - задержка между выборками (она же - скорость воспроизведения).

Байт резерва, как и раньше, предназначен для обратного воспроизведения. Для включения этого эффекта надо заменить операцию NOP в этом байте на ADD HL,DE, а INC HL в строке 200 на DEC HL.

Строки 120 и 130 нужны только для временной синхронизации с подпрограммой записи звука (по адресу 124 находится команда RET).

Всё сказанное относительно задержки между выборками и распределения памяти для подпрограмм, использующих первый способ записи, справедливо и для подпрограмм, использующих второй способ.

Особенностью подпрограмм второго типа является то, что длительность записываемого звука зависит не только от объёма выделенной памяти, но и от самого звука. Чем меньше перепадов уровней в записываемом звуке (иными словами - чем меньше его средняя частота), тем более продолжительный звук можно записать. Например, если записывать одну "тишину", то

памяти хватит на несколько минут!

Вы можете произвольно менять адрес и длину запоминаемого звука, но при этом необходимо быть предельно осторожным и не испортить жизненно важные области ОЗУ. Такие, как стек или системные переменные бейсика.

Звук, записанный этими подпрограммами можно сохранить на ленте или диске с помощью обычных команд записи, а затем загружать по мере надобности.

Так как даже не очень длинный звуковой фрагмент занимает в памяти много места, возникает естественное желание как-нибудь его сократить. Спешу Вас обрадовать - это возможно! Звуковые данные очень хорошо компрессируются по любому алгоритму. Для этого можно использовать даже экранные компрессоры. Неплохо было бы их немного модифицировать для такого использования, но, если Вы не сможете это сделать, дам несколько советов. Во-первых, без изменения компрессора можно сжимать только те файлы, размер которых меньше или равен 6912 байтам (если какой-либо компрессор отказывается принимать файлы с длиной отличной от 6912 байт, то дополните их до данного размера нулями). Во-вторых, обычно компрессоры при распаковке картинки помещают её прямо в область экрана. Поэтому, придется данные перемещать оттуда в их "родной" адрес. Это можно сделать с помощью следующей подпрограммы:

```
10          LD    HL,16384    ;
20          LD    DE,16385    ; очистка
30          LD    BC,6911     ;
40          LD    (HL),L      ; экрана
50          LDIR              ;
60          CALL  DECOMP      ; вызов декомпрессора
70          LD    HL,16384    ; HL=откуда перемещать
80          LD    DE,ADDR     ; DE=куда перемещать
90          LD    BC,LENGTH   ; BC=сколько перемещать
100         LDIR              ; перемещение данных
110         CALL  3435        ; вызов подпрограммы CLS
120         RET               ; возврат
```

Данные будут на экране такое короткое время, что Вы, скорее всего, этого и не заметите (при достаточно быстрой процедуре декомпрессии).

И в заключение приведу текст программы на бейсике, загружающей и воспроизводящей звук с оптимальными параметрами:

```
10 CLEAR VAL "24899": PRINT "Wait please…"
20 FOR A=VAL "24900" TO VAL "24972": READ D: POKE A,D: NEXT A
30 CLS: PRINT "Press any key, then speak…": PAUSE NOT PI
40 CLS: RANDOMIZE USR VAL "24900"
50 CLS: PRINT "Press any key to replay…": PAUSE NOT PI
60 CLS: RANDOMIZE USR VAL "24934": GOTO VAL "30"
70 DATA 243, 33, 141, 97, 17, 114, 158, 6, 8, 203, 38, 219, 254, 203, 119, 40, 2,
  203, 198, 14, 1, 13, 32, 253, 16, 239, 35, 27, 122, 179, 32, 231, 251, 201
80 DATA 243, 33, 141, 97, 17, 114, 158, 0, 62, 7, 6, 8, 230, 239, 203, 6, 48, 2,
  246, 16, 211, 254, 14, 1, 13, 32, 253, 16, 239, 35, 27, 79, 122, 179, 121,
32, 229, 251, 201
```

Если Вы захотите включить воспроизведение "наоборот", вставьте в эту программу, следующую строку:

```
45 POKE VAL "24941",VAL "25": POKE VAL "24963",VAL "43"
```

Для отключения этого эффекта удалите данную строку и перезапустите программу.

Непривычная запись чисел в этой программе приводит к экономии памяти. Так, выражение "24900" занимает 11 байт, а "VAL "24900"" - только 8 ("NOT PI" равняется нулю).

4.7. Реверберация.

Говоря человеческим языком, реверберация - это эхо.

Программа, реализующая этот интересный эффект по существу является воссоединением обоих фрагментов первого типа из предыдущей главы. То есть, эта программа одновременно воспроизводит звук и записывает на его место новый.

Вот она:

```
10          DI                ; запрет прерываний
20 LOOP1    LD    HL,25000    ; HL=адрес буфера
```

```
30          LD    DE,2560      ; DE=задержка эха
40 LOOP2    LD    B,8          ; B=счетчик битов
50 LOOP3    XOR   A            ; A=цвет бордюра (0)
60          RRC   (HL)         ; скроллинг данных через флаг CY
70          JR    NC,NOOUT     ; если CY=0, то перейти на NOOUT
80          OR    16           ; установка бита D4 регистра A
90 NOOUT    OUT   (254),A      ; вывод A в порт 254
100         RES   7,(HL)       ; сброс бита D7 в памяти
110         XOR   A            ; A=0 (опрос всей клавиатуры)
120         IN    A,(254)      ; ввод значения из порта 254
130         BIT   6,A          ; проверить бит магнитофона
140         JR    Z,NOINP      ; если 0, то перейти на NOINP
150         SET   7,(HL)       ; установить бит D7 в памяти
160 NOINP   CPL                ; клавиша
170         AND   31           ;   нажата?
180         JR    NZ,EXIT      ; если да, то перейти на EXIT
190         LD    C,2          ; C=задержка
200 PAUSE   DEC   C            ; C=C-1
210         JR    NZ,PAUSE     ; если C<>0, то цикл
220         DJNZ  LOOP3        ; продолжить цикл обработки байта
230         INC   HL           ; HL=HL+1
240         DEC   DE           ; DE=DE-1
250         LD    A,D          ; DE=
260         OR    E            ;    0?
270         JR    NZ,LOOP2     ; если нет, то перейти на LOOP2
280         JR    LOOP1        ; перейти на LOOP1
290 EXIT    EI                 ; разрешение прерываний
300         RET                ; возврат
```

Эта подпрограмма будет работать до тех пор, пока Вы не нажмете какую-либо клавишу. А так как она очень чувствительна к подобного рода действиям, вызывать её лучше с помощью маленькой программки на бейсике:

```
10 FOR I=1 TO 10: NEXT I
20 RANDOMIZE USR A
```

- где A - адрес подпрограммы реверберaтора.

Если Вы хотите, чтобы независимо от местоположения подпрограммы буфер находился сразу за ней, то введите следующие строки:

```
20          LD    HL,BUFFER
310 BUFFER  NOP                ; здесь располагается буфер
```

В строке 190 содержится значение задержки между выборками сигнала. Меняя, его Вы можете настраивать соотношение "качество звука - объём памяти".

Изменяя значение в строке 30 можно менять задержку эха.

И последнее: если Вы будете вводить звук с выхода какого-либо устройства, то эхо будет звучать один раз. Если же - с помощью микрофона, то Вы можете установить количество повторений, регулируя громкость воспроизводимого звука и уровень записи магнитофона. Если хотя бы один из этих параметров будет ниже необходимого уровня, то эхо прозвучит один раз. Чем выше эти параметры, тем большее число повторений Вы услышите (их громкость будет, как в жизни от раза к разу уменьшаться). Однако не следует увлекаться. Если Вы переусердствуете, то программа самовозбудится и все что она будет воспроизводить, впоследствии будет звучать на фоне ужасно противного шума. Если это произошло, выключите громкость или микрофон на несколько секунд, и все вернется на свои места.

4.8. Синтезирование речи.

Синтезировать речь можно просто записывая и воспроизводя её как любой другой звук с помощью подпрограмм из главы 4.6, но обычно действуют иначе. Сначала в компьютер вводятся все фонемы определенного языка, затем у пользователя запрашивается фраза, и, наконец, компьютер последовательно воспроизводит входящие в нее фонемы. Этот способ позволяет значительно экономить память и воспроизводить абсолютно любые фразы. Вот программа, позволяющая спектруму разговаривать:

```
10 CLEAR 25343: LET L=3
```

```
20 PRINT "Wait please…"
30 FOR A=25344 TO 25433: READ D: POKE A,D: NEXT A
40 FOR A=0 TO 25: POKE 25437,A*L+100
50 CLS: PRINT "Press any key, then say '";CHR$ (A+65);"'": PAUSE 0
60 RANDOMIZE USR 25344: NEXT A
70 CLS: INPUT "Enter phrase:"; LINE A$: PRINT A$
80 FOR A=1 TO LEN A$
90 IF A$(A)=" " THEN POKE 25433+A,32:NEXT A
100 IF A$(A)>="a" THEN LET A$(A)=CHR$(CODE A$(A)-32)
110 POKE 25433+A,(CODE A$(A)-65)*L+100: NEXT A: POKE 25433+A, 0
120 RANDOMIZE USR 25406: GOTO 70
130 DATA 243, 33, 0, 0, 17, 0, L, 6, 8, 203, 38, 219, 254, 203, 119, 40, 2, 203,
  198, 16, 244, 35, 27, 122, 179, 32, 236, 251, 201
140 DATA 243, 33, 0, 0, 17, 0, L, 62, 7, 6, 8, 230, 239, 203, 6, 48, 2, 246, 16,
  211, 254, 16, 244, 35, 27, 79, 122, 179, 121, 32, 234, 251, 201
150 DATA 33, 90, 99, 126, 183, 200, 254, 32, 32, 7, 6, 5, 118, 16, 253, 24, 8, 50,
  32, 99, 229, 205, 29, 99, 225, 35, 24, 231
```

Поясню некоторые строки:

30 - считываются в память программы в кодах
40…60 - записываются фонемы с микрофона
70 - ввод фразы
80…110 - фраза заносится в память в специальном формате
120 - вызывается подпрограмма воспроизведения
130…150 - подпрограммы в кодах

После запуска программа попросит немного подождать, считывая кодовую часть, затем Вы должны будете последовательно произнести в микрофон 26 фонем. После этого Вы можете вводить любые фразы и наслаждаться их звучанием. При вводе этих фраз допустимы прописные и строчные латинские буквы и пробелы. Как будут звучать все остальные символы никому

неизвестно. Ни в коем случае не стоит вводить знак "@", иначе программа просто "зависнет".

Эта программа написана как пример, поэтому не содержит дополнительных сер- висных возможностей (например, изменение отдельной фонемы), но Вы свободно можете их добавить.

Для изменения длительности отдельной фонемы смените в десятой строке значение переменной L (в данном случае оно может быть от 1 до 6).

Если Вам не удалось с первого раза произнести все фонемы с удовлетворительным качеством, то можете попытаться еще раз, остановив программу с помощью клавиши BREAK и выполнив команду GO TO 40. Когда фонемы будут записаны, Вы можете сохранить их на ленте, набрав команду SAVE "name" CODE 25600,6656*L (число 6656 получилось путём умножения количества фонем (26) на 256). Лучше всего, если перед этим блоком будет записана вышеприведенная программа со следующей строкой:

```
35 LOAD "name" CODE 25600: GOTO 70
```

Записать эту программу можно с помощью команды SAVE "SPEAKER" LINE 10.

К сожалению, 26 фонем - это маловато даже для английского языка, не говоря уже про русский. Их количество можно конечно увеличить, но при этом увеличится и объём занимаемой ими памяти. Тут уж ничего не сделаешь.

Если Вы хотите чтобы Ваш ZX-Spectrum говорил по-русски, то Вы должны подключить русский шрифт (подробнее см. в [1]) и увеличить число фонем. А также сменить все сообщения, выводимые на экран.

Количество русских фонем можно значительно сократить, если не использовать мягкие согласные (в компьютерном исполнении их мягкость практически не слышна) и гласные "Е", "Ё", "Ю" и "Я" (вместо них вводите "Й" и, соответственно, "Э", "О", "У" и "А"). Твердый знак также заменяйте буквой "Й".

И в заключение один совет. Гораздо приятнее слышать нормальную человеческую речь, чем слова, выговариваемые в соответствии с правилами грамматики. Поэтому вводите слова как слышите: Sinclair - sinkler, компьютер - кампйутер.

4.9. Звук на прерываниях.

У всех подпрограмм, приведенных в предыдущих главах, есть один общий недостаток: на время их звучания выполнение основной программы приостанавливается. Это можно исправить, хотя и с трудом, используя второй режим прерываний. Те, кто знает, что это такое, могут, смело пропустить следующие несколько абзацев.

Как уже говорилось раньше, пятьдесят раз в секунду процессор получает сигнал о необходимости вызова прерывания. При этом он вызывает некоторую подпрограмму, после чего продолжает обрабатывать основную программу.

Существует три различных режима прерываний 0, 1 и 2, которые выбираются командами IM 0, IM 1 и IM 2. Стандартный режим имеет номер 1. О нём уже было рассказано в главе 3, но на всякий случай напомню, что в этом режиме в качестве обработчика прерываний используется подпрограмма ПЗУ по адресу 56 (#38), которая следит за клавиатурой и счётчиком времени. Режим 0 нам не интересен, так как в ZX-Spectrum он аналогичен режиму 1. А вот режим 2 и есть самое интересное!

Во втором режиме прерываний пятьдесят раз в секунду происходит следующее: Микропроцессор считывает с шины данных байт, называемый вектором прерывания. Он передается в младший байт шины адреса, а в старший байт записывается содержимое регистра I. По полученному таким образом адресу процессор считывает из памяти два байта, которые интерпретируются как адрес подпрограммы обработки прерывания.

ZX-Spectrum устроен так, что вектор прерывания обычно равен 255 (#FF), но некоторые внешние устройства, например, AMX-mouse, могут генерировать другие вектора. Кроме того, в некоторых некачественных спектрумах вектор прерывания может изменяться абсолютно случайным образом.

Исходя из всего вышесказанного, можно предложить следующую последовательность действий для установки собственной подпрограммы обработки прерываний:

1. запретить прерывания
2. записать в память адрес обработчика прерываний
3. задать в регистре I старший байт адреса указателя на обработчик
4. установить второй режим прерываний
5. разрешить прерывания

А для возврата к стандартному режиму обработки прерываний нужно выполнить такие действия:

1. запретить прерывания
2. записать в регистр I число 63
3. установить первый режим прерываний
4. разрешить прерывания

Так как вектор прерывания может изменяться, вместо записи двух байтов по определенному адресу выстраивается целая таблица размером 257 байт с таким расчётом, чтобы при любом значении вектора считывался один и тот же адрес. Понятно, что для этого все байты таблицы должны быть одинаковыми.

При составлении процедур обработки прерываний следует придерживаться некоторых правил. Во-первых, обработчик прерывания должен выполняться за достаточно короткий промежуток времени. Во-вторых, все регистры, используемые в обработчике перед возвратом должны принять значения, находившиеся в них до вызова прерывания. В связи с этим не рекомендуется обращаться к подпрограммам ПЗУ, по крайней мере, до тех пор, пока Вы не знаете совершенно точно, какие в них используются регистры, и какие системные переменные при этом могут быть изменены. Вызов подпрограмм ПЗУ не желателен ещё и потому, что некоторые из них разрешают прерывания, что совершенно не допустимо во избежание вызова обработчика из самого себя. Любой обработчик должен работать при запрещенных прерываниях. Однако, использовать команду DI в самом начале процедуры не обязательно, так как это делается автоматически и Вам надо позаботиться только о разрешении прерываний перед выходом.

Если Вы не хотите лишаться возможностей, предоставляемых стандартным обработчиком прерываний, можете использовать в своей подпрограмме команду RST 56. А при использовании

прерываний в бейсик-программах это просто необходимо, иначе будет заблокирована клавиатура.

Теперь разберемся, чем нам могут быть полезны прерывания.

Если задать некоторую последовательность звуков и в каждом прерывании воспроизводить короткую её часть, то получится довольно хороший эффект параллельного с программой звука. Причем каждая часть должна быть, действительно, очень короткой, иначе толку от прерываний не будет.

Условимся, что последовательность звуков будем задавать блоком данных в следующем формате: для каждой ноты в блоке данных должно быть по два байта. Первый байт - частота (1…253), а второй байт - длительность (0…255). Кроме того, могут встречаться такие контрольные коды:

0 - переключение тон/шум
254 - начало цикла
255 - конец блока

Изначально программа настроена на вывод чистого тона, но, если Вам необходимо получить шум, Вы можете переключить её на воспроизведение шума, вставив в данные байт, равный 0. Для повторного переключения на тон байт 0 должен встретиться еще раз.

Когда программа встретит байт 255, вывод звука либо прекратится, либо вся последовательность повторится снова - в зависимости от заданного числа повторений.

Если в блоке данных встретится код 254, то при очередном повторении эффект начнется не с начала, а с места, где этот код встретился.

Теперь сама программа. Она является законченным комплексом и может быть оттранслирована ассемблером без малейших изменений:

```
10          ORG   60000
20          JP    SINIT       ; подключение прерываний
30          JP    SSTOP       ; отключение прерываний
40          JP    NEWFX       ; инициализация эффекта
50 MUTE     LD    (COUNT),A   ; "заглушка"
60          RET
70 SINIT    XOR   A
80          LD    (COUNT),A
90          LD    A,24        ; код команды JR
100         LD    (65535),A
110         LD    A,195       ; код команды JP
120         LD    (65524),A
130         LD    HL,INTR     ; HL=адрес обработчика
140         LD    (65525),HL
150         LD    HL,65024
160         LD    DE,65025
170         LD    BC,256
180         LD    (HL),255    ; адрес прерывания - 65535
190         LD    A,H
200         LDIR              ; заполнение таблицы
210         DI
220         LD    I,A
230         IM    2
240         EI
250         RET
260 SSTOP   DI                ; отключение прерываний
270         LD    A,63
280         LD    I,A
290         IM    1
300         EI
310         RET
320 NEWFX   DI                ; инициализация эффекта
330         LD    (COUNT),A
340         XOR   A
350         LD    (FLAG),A
360         LD    (ADDR),HL
370         LD    (CURADD),HL
380         EI
```

```
390          RET
400 ADDR     DEFW  0            ; начальный адрес блока данных
410 CURADD   DEFW  0            ; текущий адрес в блоке данных
420 COUNT    DEFB  0            ; количество повторений
430 FLAG     DEFB  0            ; флаг тон/шум
440 INTR     PUSH  AF           ; обработчик прерывания
450          PUSH  BC           ; сохранение регистров
460          PUSH  DE
470          PUSH  HL
480 TEST     LD    A,(COUNT)    ; A=счетчик повторений
490          OR    A            ; есть что играть?
500          JR    Z,EXIT
510          LD    HL,(CURADD); HL=текущий адрес
520 NEXT     LD    A,(HL)
530          INC   HL
540          CP    254          ; A=254? (начало цикла)
550          JR    NZ,CONT1
560          LD    (ADDR),HL    ; изменение начального адреса
570          JR    NEXT
580 CONT1    CP    255          ; A=255? (конец)
590          JR    NZ,CONT2
600          LD    HL,(ADDR)    ; восстановление начального
610          LD    (CURADD),HL ; адреса блока данных
620          LD    HL,COUNT
630          DEC   (HL)         ; уменьшение счётчика повторений
640          JR    TEST
650 CONT2    OR    A            ; A=0? (переключатель)
660          JR    NZ,CONT3
670          LD    A,(FLAG)
680          CPL                ; инвертирование A
690          LD    (FLAG),A
700          JR    NEXT
710 CONT3    LD    B,A          ; B=частота
720          LD    C,(HL)       ; C=длительность
730          INC   HL
740          LD    (CURADD),HL ; сохранение текущего адреса
750          LD    A,(FLAG)     ; A=флаг
760          OR    A            ; A=0?
770          LD    A,7          ; A=цвет бордюра
780          JR    NZ,NOISE
790 TONE     XOR   16           ; воспроизведение тона
800          OUT   (254),A
810          PUSH  BC
820 PAUSE    DJNZ  PAUSE
830          POP   BC
840          DEC   C
850          JR    NZ,TONE
860          JR    EXIT
870 NOISE    LD    HL,1000      ; воспроизведение шума
880          LD    D,A
890 NOIS2    LD    A,(HL)
900          AND   248
910          OR    D
920          OUT   (254),A
930          PUSH  BC
940 PAUS2    DJNZ  PAUS2
950          POP   BC
960          INC   HL
970          DEC   C
980          JR    NZ,NOIS2
990 EXIT     POP   HL           ; восстановление регистров
1000         POP   DE
1010         POP   BC
1020         POP   AF
1030         RST   56           ; вызов стандартного обработчика
```

```
1040         RET              ; возврат
```

Порядок использования этого пакета должен быть следующим. В начале работы нужно вызвать подпрограмму SINIT, которая включит 2-ой режим прерываний. В тот момент, когда Вам нужно получить звук надо занести в регистр HL адрес блока данных, в регистр A - число повторений и вызвать подпрограмму NEWFX. Затем, если Вам по какой-либо причине надо выключить звук или продлить его звучание, занесите в регистр A новое число повторений и вызовите подпрограмму MUTE. Кроме того, эту подпрограмму

можно использовать для включения последнего звучавшего эффекта. По окончании работы (или если в программе предусмотрены обращения к дисководу) следует вызвать подпрограмму SSTOP, которая восстановит стандартный режим прерываний.

Серия команд JP в начале пакета сделана для удобства. Благодаря ей все подпрограммы этого пакета могут вызываться по соседним адресам:

SINIT - 60000 (#EA60)
SSTOP - 60003 (#EA63)
NEWFX - 60006 (#EA66)
MUTE - 60009 (#EA69)

К сожалению, с помощью прерываний невозможно получить достаточно чистый тон. Поэтому, а также из-за совсем не музыкального формата данных, эта программа вряд-ли может использоваться для создания музыки. Но для звуковых эффектов она в самый раз.

Вот пример использования приведенной программы:

```
10           ORG   50000
20           CALL  60000
30           LD    HL,SNDFX
40           LD    A,3
50           CALL  60006
60           RET
70 SNDFX     DEFB  200,5,250,4,200,5,100,10,75,13,50,20,255
```

Строку 70 можно заменить на следующую:

```
70 SNDFX     DEFB 50,20,75,13,100,10,200,5,250,4,50,20,255
```

И последнее. Если в Вашей программе не очень много движущихся объектов на экране и длительность звукового эффекта достаточно мала (около полсекунды или меньше), то его можно использовать без всяких прерываний - задержка заметна не будет.

5. Оператор PLAY.

С этой главы начинается часть книги, посвященная музыкальному сопроцессору AY-3-8910 (или AY-3-8912), встроенному в ZX-Spectrum 128, а также во все его модификации.

При включении ZX-Spectram 128 на экране появляется меню, предлагающее на выбор несколько пунктов. Все они неплохо описаны в [2], в нашем же случае выберите пункт 128 BASIC. В этом режиме Вам становится доступен оператор PLAY, обслуживающий музыкальный сопроцессор. В общем виде формат оператора PLAY выглядит так:

```
PLAY A$[,B$,C$,D$,E$,F$,G$,H$]
```

- где A$…H$ - символьные строки, задающие музыкальную программу. Сам сопроцессор использует для управления своими тремя каналами (A, B и C) только A$, B$ и C$. Если для Вас три канала много, то Вы можете использовать только два, или даже один, задав соответствующее количество стрингов (пустой стринг также указывает на то, что канал не используется).

Остальные строки (D$…H$) предназначены для управления музыкальными инструментами, которые можно подключить к ZX-Spectram 128 через MIDI-интерфейс.

Все заданные стринги исполняются одновременно, что позволяет создавать сложные и красивые мелодии.

В стрингах используются специальные команды. Остановимся на них подробнее.

Ноты ДО, РЕ, МИ, ФА, СОЛЬ, ЛЯ и СИ текущей октавы обозначаются строчными латинскими буквами, соответственно, c, d, e, f, g, a и b (по международному стандарту). Ноты следующей октавы обозначаются прописными латинскими буквами C, D, E, F, G, A и B. Например, оператор

```
PLAY "cdefgabC"
```

проиграет гамму ДО-мажор.

Полутона задаются знаками # (диез) и $ (бемоль), стоящими перед нотой. Например, ДО-диез запишется как #c, а СИ-бемоль - $b. Чтобы проиграть гамму ДО-минор, введите

```
PLAY "cd$efg$a$bC".
```

Допустимо использование нескольких диезов или бемолей подряд. Так, ноту d можно записать как ##c.

Длительность определяется числом от 1 до 9, стоящим перед нотой (табл. 3). Установленная длительность распространяется на все последующие ноты и паузы (см. команду &). По умолчанию устанавливается длительность, равная 5 (одна четвертая).

Число	Нота
	Одиночные ноты
1	Одна шестнадцатая
2	Одна шестнадцатая с точкой
3	Одна восьмая
4	Одна восьмая с точкой
5	Одна четвертая
6	Одна четвертая с точкой
7	Одна вторая
8	Одна вторая с точкой
9	Целая
	Триоли
10	Одна шестнадцатая
11	Одна восьмая
12	Одна четвертая

Табл. 3. Длительности нот в операторе PLAY.

Кроме обычных нот Вы можете задавать триоли. Для этого используются числа от 10 до 12. Ноты триоли следуют непосредственно за числом. Например:

```
PLAY "3fed&11fed&fed"
```

Триоли не изменяют установленной длительности нот.

Кроме стандартных длительностей нот Вы можете использовать произвольные. Для этого перед нотой через символ подчеркивания ("_") ставят несколько параметров, задающих требуемую длительность. Например, ноту ДО длительностью 3/8 (1/8 +1/4) можно записать так: "3_5c".

Длительность всех последующих нот и пауз определяет последний параметр в связке.

Довольно странные эффекты получаются при соединении длительностей обычных нот и триолей (например: "3_12cde"). Поэкспериментируйте с этим. Про запись нот, вроде бы, все. Теперь о командах:

O Изменение октавы. Номер октавы задается числом от 0 до 8, следующим за командой (по умолчанию установлена октава 5). Соответствие параметра команды O музыкальным октавам приведено в таблице 4.

Число	Октава
0	?
1	Субконтроктава
2	Контроктава
3	Большая октава
4	Малая октава
5	Первая октава
6	Вторая октава
7	Третья октава
8	Четвертая октава

Табл. 4. Октавы в операторе PLAY.

Обратите внимание, что ноты ниже b октавы 1 без MIDI-интерфейса воспроизводиться будут неправильно. Также немаловажно, что хотя максимальный номер октавы равен восьми, ничто Вам не мешает использовать следующую - девятую, обозначая ноты прописными буквами.

N Разделитель параметров. Используется для отделения двух числовых параметров друг от друга. Например, когда надо задать длительность ноты сразу за установкой октавы: "O4N5d". По существу, эта команда является излишеством, так как вместо нее можно использовать пробел: "04 5d".

& Пауза. Длительность паузы устанавливается так же, как и длительность ноты (см. выше). Например: "6c&de3&de".

V Установка громкости. Громкость задается числом от 0 (минимальная - выключено) до 15 (максимальная). Например: "V10". По умолчанию установлена громкость 15.

W Программирование эффектов. Ноты могут воспроизводиться не только с фиксированной громкостью, но и со всевозможными эффектами: затуханиями, всплесками и т. д. Характер эффекта задается числом от 0 до 7, в соответствии с таблицей 5. По умолчанию установлен эффект 0.

X Временной параметр звукового эффекта. Для эффектов 0…3 параметр задает длительность действия, для 4 и 5 - период, 6 и 7 - полупериод (см. табл. 5). Значение параметра выбирается из диапазона 0…65535. По умолчанию установлено 65535.

Параметр	Эффект	Диаграмма
0	Спад, затем тихо	
1	Подъем, затем тихо	
2	Спад, затем громко	
3	Подъем, затем громко	
4	Повторяющийся спад	
5	Повторяющийся подъем	
6	Повторяющийся подъем-спад	
7	Повторяющийся спад-подъем	

Табл. 5. Программирование эффектов.

U Включение звукового эффекта. После этой команды все ноты будут воспроизводиться с эффектом, установленным командами W и X. Эффекты отключаются по завершении стринга, либо при изменении громкости (команда V).

Следующая программа продемонстрирует действие всех эффектов:

```
10 FOR A=0 TO 7
20 PLAY "UX1000W"+STR$ A+"cdef&"
30 NEXT A
```

К сожалению, в музыкальном сопроцессоре всего один генератор огибающей, поэтому Вам не удастся настроить несколько каналов на разные эффекты одновременно. Хотя одинаковые эффекты Вы можете использовать сразу в нескольких каналах, причем параметры огибающей необходимо задать только в одном из них, а в остальных достаточно указать команду U (см. выше).

T Темп исполнения. Задается числом от 60 до 240. По умолчанию устанавливается темп, соответствующий команде T120. Задать темп можно лишь для всей мелодии в целом, поэтому он определяется только в стринге канала A (в других каналах команда T игнорируется).

(.) Повтор (репризы). Музыкальная фраза, взятая в скобки, повторится два раза. Допустимо использовать скобки в скобках (до 4 вложений). Закрывающая скобка без соответствующей ей открывающей заставит музыкальную фразу повторяться бесконечно от начала стринга. Используется это, например, в басовых и ударных партиях.

H Останов оператора PLAY. Встретив эту команду в стринге любого канала оператор PLAY закончит свою работу. Команда используется, например, для выхода из "зацикленной" басовой партии, когда заканчивается основная мелодия.

!.! Комментарий. Используется для вставки пояснений. Например:

```
PLAY "!SCALE:!cdefgabC".
```

Если после комментария в стринге никаких команд нет, то закрывающий знак "!" ставить необязательно.

M Устанавливает режим работы каналов. Каждый канал может воспроизводить не только чистый тон, но и так называемый белый шум. Распределение режимов задается числом от 1 до 63, определяемым суммой кодов, соответствующих режиму каждого из каналов (табл. 6). Как и генератор огибающей, генератор шума в музыкальном сопроцессоре только один, поэтому Вам не удастся настроить несколько каналов одновременно на шум с разной частотой. По умолчанию все три канала настроены на тон (команда M7). В следующем примере канал A настраивается на тон, а канал B - на шум:

```
PLAY "M17cegbdfaCH","O3cC)".
```

Если Вы укажете команду MO, то все каналы будут молчать, а Вы услышите потрескивание.

Канал	A	B	C
Тон	1	2	4
Шум	8	16	32

Табл. 6. Режимы работы каналов.

Y Установка канала MIDI-интерфейса. Число от 1 до 16, следующее за командой, задает номер канала интерфейса, в который необходимо направить вывод музыкальных данных.

Z Передача управляющих кодов MIDI-интерфейсу. Код MIDI (число от 0 до 255) должен располагаться сразу за командой. Допустимо передавать сразу два байта. В этом случае их значения нужно отделить друг от друга пробелом или командой N.

Обратите внимание, что все команды оператора PLAY должны быть набраны прописными буквами!

В заключение приведу краткую сводку команд оператора PLAY (табл. 7) и сообщения об ошибках, которые могут возникать во время его выполнения (табл. 8).

Команда	Действие
a-g, A-G	Ноты текущей и следующей октав
H	Останов оператора PLAY
M	Устанавливает режимы работы каналов (1-63)
N	Разделитель параметров
O	Устанавливает текущую октаву (0-8)
T	Устанавливает темп (60-240)
U	Включает звуковые эффекты
V	Устанавливает громкость (0-15)
W	Задает вид звукового эффекта (0-7)
X	Задает длительность звукового эффекта (0-65535)
Y	Устанавливает канал MIDI-интерфейса (1-16)
Z	Передает данные MIDI-интерфейсу
1-12	Длительность нот или пауз
!...!	Комментарий
(...)	Репризы
$	Бемоль
#	Диез
&	Пауза
_	Соединитель длительностей

Табл. 7. Команды оператора PLAY.

Код	Сообщение	Значение
D	Too many brackets	Превышено число вложений скобок
K	Invalid note name	Встречена неопознанная команда
L	Number too big	Указано недопустимое значение параметра
M	Note out of range	Нота вышла за диапазон муз. сопроцессора
N	Out of range	Числовой параметр слишком мал или велик
O	Too many tied notes	Слишком много длительностей связано знаком "_"

Табл. 8. Ошибки оператора PLAY.

5.1. Создание эффектов на PLAYе.

С помощью оператора PLAY можно создавать очень интересные звуковые эффекты. Причем, делается это значительно проще, чем в случае с BEEPом. Количество возможных эффектов ограничено только фантазией программиста. Чаще всего в них используются громкостные эффекты, многоголосие и генератор шума.

Так как количество эффектов в данном случае неимоверно велико, не буду описывать каждый из них (принципы те же, что и в главе 2.1), а приведу несколько примеров с необходимыми комментариями.

Пример первый:

```
10 PLAY "X16384W0O6U9B"
```

- воспроизводит плавно затухающую ноту.

Пример второй:

```
10 PLAY "O6X1000W3U1BX16384W0 9B"
```

- модификация предыдущего эффекта. Громкость ноты сначала возрастает от минимальной до максимальной (мягкая атака), а затем плавно затухает.

Пример третий:

```
10 PLAY "T240W0X3000UO8(B&c&)"
```

- этот эффект практически полностью аналогичен предыдущим, но звучит совсем по-другому.

Пример четвертый:

```
10 PLAY "O4X16384W3U9B"
```

- здесь используется эффект, который в музыке называется мягкой атакой звука.

В этих четырех эффектах применяется чистый тон, но Вы можете использовать и шум, вставив команду M8, или смесь тона с шумом (команда M9). Следующий пример такой:

```
10 LET A$="T240"
20 FOR A=1 TO 8
30 LET A$=A$+"O"+STR$ A+" 1CDEFGAB"
40 NEXT A: PLAY A$
```

В этом эффекте сначала подготавливается музыкальная программа, а затем проигрывается. Её можно было бы задать прямо в операторе PLAY, но тогда это выглядело бы ужасно (несколько строк однотипного текста!). Если же воспроизводить её одновременно с созданием (в цикле), то появятся нежелательные паузы.

Перед заданием основных данных в переменную A$ заносится команда T240, что позволяет ускорить эффект. Попробуйте кроме T240 добавить M8 или M9.

Следующий пример:

```
10 LET A$="T240": LET B$=""
20 FOR A=1 TO 8
30 LET A$=A$+"O"+STR$ A+" 1CDEFGAB"
40 LET B$=B$+"O"+STR$ (9-A)+" 1BAGFEDC"
50 NEXT A: PLAY A$, B$
```

Этот эффект очень похож на предыдущий. Но в данном случае используются два голоса, частоты которых смещаются в противоположных направлениях. Еще один пример:

```
10 PLAY "O3 1cdefgabC","O5 1cdefgabC","O6 1cdefgabC"
```

Здесь используются все три канала, настроенные на разные частоты, что звучит довольно интересно.

В последнем примере нетрадиционно используется генератор огибающей. С его помощью формируется необычный тембр:

```
10 PLAY "W4X1UcdefgabCW7X4cdefgabC"
```

Возможности оператора PLAY не исчерпываются приведенными эффектами. Вы можете поэкспериментировать с ним.

И в заключение немного критики. Оператор PLAY, несмотря на все свои достоинства (которые принадлежат скорее музыкальному сопроцессору), имеет и недостатки. Например, он не позволяет воспроизводить ноты в режиме легато (слитно), а также очень короткие ноты.

5.2. Создание музыки на PLAYe.

С помощью оператора PLAY можно очень легко создавать неплохие трехголосые мелодии. Все его возможности описаны в главе 5, поэтому здесь я ограничусь несколькими примерами:

```
10 PLAY "T60W0X10000U3g4g1CCb5b3f4f1aag5g3eag1g#f5f3feda5g"
10 PLAY "T180W0X10000U7c6e3f9&7c6e3f&O4efefe5g&",
   "O3U3c&&&e&&f&E&F&6G3c&&&e&&f&O4EFEFE5G3c"
```

Мелодию можно разнообразить с помощью ударных инструментов. Вот, например, всем известная песенка "В траве сидел кузнечик" в стиле панк-рок:

```
10 PLAY "T240M35fcfcfee&ececeff&fcfcfee&ececefH",
   "V14O4&f&c&e&c&e&c&f&c&f&c&e&c&e&c&f", "W0X2000Uc&)"
```

Несмотря на то, что с помощью оператора PLAY можно создавать неплохую музыку и эффекты, самые красивые варианты получаются в кодах. Поэтому настоятельно Вам советую изучить следующую главу.

6. Управление музыкальным сопроцессором.

Музыкальный сопроцессор AY-3-8910 (AY-3-8912) позволяет генерировать трехканальный звук с изменяемой громкостью и шумовыми эффектами. Эта микросхема содержит в себе шестнадцать регистров, управляющих звуком. Регистры обозначаются R0…R15.

Выбор регистра осуществляется путем записи его номера в порт 65533 (#FFFD), а затем чтением содержимого выбранного регистра из этого же порта, либо записью нового значения выбранного регистра в порт 49149 (#BFFD). Выбрав номер регистра один раз, Вы можете выполнять сколько угодно считываний или записей. И только когда требуется доступ к другому регистру надо сменить содержимое порта 65533.

После задания необходимых параметров сопроцессор начинает генерировать звук, освобождая Z-80 для выполнения других операций.

Все временные интервалы в микросхеме AY получаются путем деления его тактовой частоты, равной 1.7734 мГц, на определенное число.

6.1. Регистры.

R0 - младший байт частоты канала A
R1 - старший байт частоты канала A
R2 - младший байт частоты канала B
R3 - старший байт частоты канала B
R4 - младший байт частоты канала C
R5 - старший байт частоты канала C

Требуемая частота любого канала получается путём деления тактовой частоты сопроцессора на 16 с последующим делением результата на 12-ти разрядное значение, получаемое слиянием регистра младшего байта частоты и битов D0…D3 регистра старшего байта частоты. Таким образом, всего можно задать 4095 различных частот (от 27 Гц до 110 кГц). Естественно, самых высоких Вы не услышите. Обратите внимание, что разница между двумя соседними значениями в области низких частот составляет доли герца, а в области высоких частот достигает нескольких килогерц.

Младший и старший байты, а также частоту можно рассчитать по следующим формулам:

```
Hi=INT(INT(110830/Fq+.5)/256)
Lo=INT(110830/Fq+.5)-Hi*256
Fq=110830/(Lo+256*Hi)
```

- где Hi - старший байт, Lo - младший байт, а Fq - частота в Гц. Обратите внимание, что в большинстве случаев заданную частоту невозможно точно передать музыкальному сопроцессору. В таблице 9 приведены значения регистров тона для всех возможных нот.

нота \ октава	СК	К	Б	М	1	2	3	4	5	6
ДО		3389	1694	847	424	212	106	53	26	13
ДО-диез		3199	1599	800	400	200	100	50	25	12
РЕ		3019	1510	755	377	189	94	47	24	
РЕ-диез		2850	1425	712	356	178	89	45	22	
МИ		2690	1345	672	336	168	84	42	21	
ФА		2539	1269	635	317	159	79	40	20	
ФА-диез		2396	1198	599	300	150	75	37	19	
СОЛЬ		2262	1131	565	283	141	71	35	18	
СОЛЬ-диез		2135	1067	534	267	133	67	33	17	
ЛЯ	4030	2015	1008	504	252	126	63	31	16	
ЛЯ-диез	3803	1902	951	476	238	119	59	30	15	
СИ	3590	1795	898	449	224	112	56	28	14	

Табл. 9. Значения регистров тона.

R6 - задает частоту шума

Требуемая частота шума получается путем деления тактовой частоты на 256 с последующим делением результата на 5-ти разрядное значение, расположенное в пяти младших разрядах (D0…D4) регистра R6. Всего можно задать 31 различную частоту (от 223 Гц до 7 кГц).

Значение регистра R6 и частоту шума можно рассчитать по следующим формулам:

```
R=INT(6927.3437/Fq+.5)
Fq=6927.3437/R
```

- где R - значение R6, а Fq - частота в Гц.

Все сказанное про генераторы тона, также справедливо и для генератора шума.

R7 - управление микшером и вводом/выводом

Отдельные биты этого регистра используются в различных целях:

D0 - тон канала A
D1 - тон канала B
D2 - тон канала C
D3 - шум канала A
D4 - шум канала B
D5 - шум канала C
D6 - порт A: 0 - ввод/1 - вывод
D7 - порт B: 0 - ввод/1 - вывод (только AY-3-8910)

Ноль в битах D0…D5 указывает на то, что функция разрешена. Таким образом, каждый канал можно настраивать на три различных состояния: тон, шум и тон+шум. Порты ввода/вывода содержащиеся в музыкальном сопроцессоре в ZX-Spectrum не используются, однако, к ним можно подключить дополнительные джойстики, принтер, MIDI-интерфейс или что-нибудь подобное. Порт B существует только в AY-3-8910.

R8 - управление амплитудой канала A
R9 - управление амплитудой канала B
R10 - управление амплитудой канала C

Биты D0…D3 этих регистров устанавливают громкость для каждого из каналов (от 0 до 15). Если в одном (или нескольких) из них установлен бит D4, то значение в младших битах игнорируется, а амплитудой данного канала управляет генератор огибающей (см. дальше).

R11 - младший байт периода огибающей
R12 - старший байт периода огибающей

Требуемый период огибающей получается путем деления тактовой частоты сопроцессора на 256 с последующим делением результата на 16-ти разрядное значение, получаемое слиянием регистров R11 и R12. Всего можно задать 65535 различных периодов. Значения регистров R11 и R12, а также частоту и период огибающей можно рассчитать по следующим формулам:

```
Hi=INT(INT(6927.3437/Fq+.5)/256)
Lo=INT(6927.3437/Fq+.5)-Hi*256
Fq=6927.3437/(Lo+256*Hi)=1/Pd
Pd=(Lo+256*Hi)/6927.3437=1/Fq
```

- где Lo и Hi - значения регистров R11 и R12, соответственно, Fq - частота огибающей в Гц, а Pd - период огибающей в секундах.

R13 - управление формой огибающей

Требуемая форма огибающей задается битами D0…D3 регистра R13 следующим образом:

D0 – затухание
D1 – чередование
D2 – нарастание
D3 - продолжение

В таблице 10 приведены все возможные комбинации этих битов.

D3 D2 D1 D0	Значение R13	Эффект	Диаграмма
0 0 X X 1 0 0 1	0…3 9	Спад, затем тихо	
0 1 X X 1 1 1 1	4…7 15	Подъем, затем тихо	
1 0 0 0	8	Повторяющийся спад	
1 0 1 0	10	Повторяющийся спад-подъем	
1 0 1 1	11	Спад, затем громко	
1 1 0 0	12	Повторяющийся подъем	
1 1 0 1	13	Подъем, затем громко	
1 1 1 0	14	Повторяющийся подъем-спад	

Табл. 10. Значения регистра R13.

R14 - порт ввода/вывода A
R15 - порт ввода/вывода B

Как уже говорилось выше, порты ввода/вывода большого значения не имеют и на генерируемый звук не влияют.

6.2. Программирование.

Все звуковые эффекты и музыка программируются путем постоянной смены значений регистров с необходимыми задержками. Осуществить это может, например такая подпрограмма:

```
10          LD    HL,60000    ; НЬ=адрес данных
20 LOOP     LD    A,(HL)      ; A=байт данных
30          INC   HL          ; HL=HL+1
40          CP    255         ; A=255?
50          RET   Z           ; если да, то возврат
60          CP    16          ; A=16?
70          JR    NZ,REG      ; если нет, то перейти на REG
80          LD    B,(HL)      ; B = длительность паузы
90 PAUSE    HALT              ; ожидание прерывания
100         DJNZ  PAUSE       ; цикл
110         JR    CONT        ; перейти на CONT
120 REG     LD    BC,65533    ; BC=адрес порта регистра
130         OUT   (C),A       ; записать номер регистра
140         LD    B,191       ; BC=адрес порта данных
150         LD    A,(HL)      ; A=значение регистра
160         OUT   (C),A       ; записать данные в регистр
170 CONT    INC   HL          ; HL=HL+1
180         JR    LOOP        ; переход к началу
```

Эта программа довольно примитивна, и для воспроизведения музыки не очень подходит, но для создания простых эффектов - в самый раз. Перед её запуском не забудьте подготовить блок данных по адресу 60000, состоящий из пар данных и заканчивающийся числом 255. Первым значением в каждой паре должен быть номер регистра, а вторым - число, которое нужно записать в этот регистр. Кроме того, если первое значение равно 16, то второе интерпретируется как задержка (в пятидесятых долях секунды).

Для воспроизведения музыки используются гораздо более сложные подпрограммы, как правило работающие во втором режиме прерываний. Данные для этих подпрограмм обычно хранятся в значительно более удобной форме, раздельно для каждого из трёх каналов. Привести полностью хотя бы простейший пример такой подпрограммы в этой книге не представляется возможным из-за его сложности, но, если Вы умеете обращаться с ассемблером, Вам ничего не стоит сочинить такую подпрограмму, а я могу подсказать, как это сделать.

Сначала разберёмся с форматом данных. Я предлагаю следующую систему: мелодия будет задаваться тремя (по числу голосов) основными блоками данных. Так как практически любая мелодия состоит из одинаковых фрагментов, повторяющихся в разном порядке, логично было бы в главных блоках задавать не саму мелодию, а адреса таких фрагментов. Эти фрагменты обычно называют паттернами (pattern – трафарет, шаблон).

Итак, в главных блоках могут содержаться следующие двухбайтовые значения:

65535 (#FFFF) - конец мелодии
0 (#0000) - начало цикла

addr (#XXXX) - адрес следующего паттерна

Код "начало цикла" (0) отмечает место, с которого начнет воспроизводиться мелодия при повторении.

Теперь займемся паттернами. Не вдаваясь в теорию, приведу разработанный мной формат:

128	(#80)	- конец паттерна
129,n	(#81,#XX)	- установить длительность n
130,n	(#82,#XX)	- шум с частотой n(0…31)
131,n1,n2	(#83,#XX,#XX)	- шум с частотой n (0…31) + нота (0…100)
132,n1,n2	(#84,#XX,#XX)	- прямое задание частоты тона (0…4095)
133,addr	(#85,#XXXX)	- задание блока изменения частоты тона
134,addr	(#86,#XXXX)	- задание блока изменения частоты шума
135,addr	(#87,#XXXX)	- задание блока изменения громкости
136,n1,n2,n3	(#88,#XX,#XXXX)	- управление генератором огибающей
0…100	(#00…#64)	- ноты от ЛЯ субконтроктавы

Поясню некоторые коды:

129 - Как Вы, наверное, заметили, при задании ноты её длительность не указывается. Дело в том, что используется длительность, установленная заранее с помощью этой команды. Длительность измеряется в пятидесятых долях секунды.

131 - С помощью этого кода Вы можете одновременно воспроизводить тон и шум.

133,134,135 - Эти коды задают дополнительные блоки данных, указывающие способ изменения частоты тона, частоты шума и громкости на протяжении звучания ноты. Если после кодов 133 или 134 вместо адреса блока находится 0, то изменение частот отключается. Число от 0 до 15 после кода 135 указывает на необходимость поддержания постоянного уровня громкости, соответствующего этому числу.

136 - Этот код управляет генератором огибающей. После него должно находиться однобайтовое число от 0 до 7, указывающее на форму огибающей в соответствии с таблицей 5 и двухбайтовое число, задающее период изменения огибающей.

0…100 - Эти коды задают ноты. Код 0 соответствует ноте ЛЯ субконтроктавы, 1 - ЛЯ#, 2 - СИ и т. д.

В блоке, описывающем изменение частоты тона, используются следующие значения:

128 (#80) - конец блока

-127… 127 (#81.. .#7 9) - смещения частоты

Блок изменения частоты шума будет задаваться в следующем формате:

128 (#80) - конец блока

-31…31 (#E1.. .#1F) - смещения частоты

А блок изменения громкости пусть задается так:

128 (#80) - конец блока

0… 15 (#00.. .#0F) - значения громкости

Вполне логично было бы сделать эту процедуру работающей во втором режиме прерываний. Исходя из этого, а также из предложенного формата данных можно понять, что для каждого из каналов понадобится массив переменных. Я предлагаю следующий его формат:

смещение	*размер*	*значение*
0	*2*	*начальный адрес основного блока данных*
2	*2*	*текущий адрес в основном блоке данных*
4	*2*	*начальный адрес блока изменения частоты тона*
6	*2*	*текущий адрес в блоке изменения частоты тона*
8	*2*	*начальный адрес блока изменения частоты шума*
10	*2*	*текущий адрес в блоке изменения частоты шума*
12	*2*	*начальный адрес блока изменения громкости*
14	*2*	*текущий адрес в блоке изменения громкости*
16	*2*	*текущий адрес в текущем паттерне*
18	*1*	*значение текущей длительности*
19	*1*	*счётчик длительности*

смещение	*размер*	*значение*
20	*1*	*число оставшихся повторений*

Кроме того, понадобится ещё по три байта на каждый канал для хранения темпа, счётчика темпа и флага разрешения звука.

В этот пакет будут входить следующие процедуры:

SINIT - Инициализация таблиц, подключение второго режима прерываний.

SSTOP - Выключение сопроцессора, восстановление стандартного режима прерываний.

SNEW - Запуск всех трех каналов. При вызове этой подпрограммы в регистрах HL, DE и BC должны быть адреса главных блоков данных для каналов A, B и C, соответственно. В регистре A должно находиться число повторе ний мелодии от 1 до 254 или 255, если Вы хотите, чтобы мелодия повто рялась бесконечно.

SNEWA - Запуск канала A. В регистре HL должен быть адрес блока данных, а в регистре A - число повторений (аналогично SNEW). Работу каналов B и C эта процедура не затрагивает.

SNEWB - Запуск канала B. Все аналогично SNEWA.

SNEWC - Запуск канала C. Все аналогично SNEWA.

MUTE - Запрет/разрешение работы. В регистре A должен быть номер канала от O до 2 или 3, если Вы обращаетесь ко всем каналам. В регистре B должен быть код режима работы: 0 - остановка, 1 - беззвучная работа, 2 -воспроизведение.

STATUS - Получение состояния канала. В регистре A должен быть номер канала от 0 до 2. По возвращению из процедуры STATUS регистр A содержит код режима работы выбранного канала (аналогично B в MUTE).

TEMPO - Установка темпа. В регистре A должен быть номер канала от 0 до 2 или 3, если Вы обращаетесь ко всем каналам. В регистре B должно находиться значение темпа. Итак, начало пакета может выглядеть так:

```
10          ORG   60000
20          JP    SINIT
30          JP    SSTOP
40          JP    SNEW
50          JP    SNEWA
60          JP    SNEWB
70          JP    SNEWC
80          JP    MUTE
90          JP    STATUS
```

Процедура TEMPO может выглядеть так:

```
100 TEMPO   DI
110         PUSH  HL
120         PUSH  AF
130         LD    HL,TEMPS    ; HL=указатель на темпы
140         CP    A,3
150         JR    Z,TEMP3
160         ADD   A,L
170         LD    L,A
180         JR    NC,TEMP1
190         INC   H
200 TEMP1   LD    (HL),B
210         INC   HL
220         INC   HL
230         INC   HL
240         LD    (HL),B
250 TEMP2   POP   AF
260         POP   HL
270         EI
280         RET
290 TEMP3   PUSH  DE
300         LD    D,6
310 TEMP4   LD    (HL),B
320         INC   HL
330         DEC   D
```

```
340          JR    NZ,TEMP4
350          POP   DE
360          JR    TEMP2
```

Раз уж пошли обращения к данным, надо привести строки с их описанием:

```
370 CHAN_A   DEFS  21                     ; массив переменных для канала A
380 CHAN_B   DEFS  21                     ; массив переменных для канала B
390 CHAN_C   DEFS  21                     ; массив переменных для канала C
400 MUTS     DEFS  3                      ; флаги разрешения звука
410 TEMPS    DEFS  3                      ; темпы
420 CURTS    DEFS  3                      ; счетчики темпов
430 AYREGS   DEFS  14                     ; копии регистров сопроцессора
440 ENVS     DEFB  0,4,11,13,8,12,14,10   ; формы огибающей
450 SVOLS    DEFW  #8000,#8001,#8002      ; таблицы изменения громкости
460          DEFW  #8003,#8004,#8005      ; для стандартных значений
470          DEFW  #8006,#8007,#8008
480          DEFW  #8009,#800A,#800B
490          DEFW  #800C,#800D,#800E
500          DEFW  #800F
510 NOTES    DEFW  …                      ; сюда необходимо записать все значения из
                                          ; таблицы 9 от левого верхнего угла в порядке
                                          ; сверху - вниз, справа - налево
```

Строка 430 содержит область данных, которая используется для вывода звука. Сначала в ней формируются значения всех регистров сопроцессора с помощью следующей подпрограммы:

```
520 SETAY    PUSH  HL
530          PUSH  AF
540          LD    AL,AYREGS
550          ADD   A,L
560          LD    L,A
570          JR    NC,SETAY1
580          INC   HL
590 SETAY1   LD    (HL),B
600          POP   AF
610          POP   HL
620          RET
```

Ей надо передать номер регистра в A и его значение в B. Затем содержимое этой области копируется в реальные регистры сопроцессора другой подпрограммой:

```
630 AYOUT    PUSH  HL
640          PUSH  DE
650          PUSH  BC
660          LD    HL,AYREGS+13
670          LD    D,13
680 AYOUT1   LD    BC,65533
690          OUT   (C),D
700          LD    B,191
710          LD    E,(HL)
720          OUT   (C),E
730          DEC   HL
740          DEC   D
750          JP    P,AYOUT1    ; если D>=0, то перейти на AYOUT1
760          POP   BC
770          POP   DE
780          POP   HL
790          RET
```

Любой из регистров в этой области можно также прочитать:

```
800 GETAY    PUSH  HL
810          PUSH  AF
820          LD    HL,AYREGS
830          ADD   A,L
840          LD    L,A
850          JR    NC,GETAY1
860          INC   H
870 GETAY1   LD    B,(HL)
880          POP   AF
890          POP   HL
```

```
900          RET
```

Этой подпрограмме нужно передать номер требуемого регистра в A и она вернёт его значение в B.

Строка 440 понадобится для дешифрации кода огибающей. Если любое число из таблицы 5 прибавить к метке ENVS и по полученному адресу прочитать один байт, Вы получите значение, которое надо записать в R13.

Строка 450 будет нужна для дешифрации стандартных уровней громкости. Для этого надо число от 0 до 15 умножить на 2 (можно использовать команду SLA) и прибавить к метке SVOLS. Полученное значение нужно использовать в качестве адреса блока изменения громкости.

Строка 510 будет полезна для дешифрации нот. Если код ноты (от 0 до 100) умножить на 2 (команда SLA) и прибавить к метке NOTES, а по полученному адресу считать двухбайтовое число, Вы получите значения младшего и старшего регистров частоты.

Теперь займёмся основными процедурами:

```
910 STATUS  PUSH  HL
920         LD    HL,MUTS
930         ADD   A,L
940         LD    L,A
950         JR    NC,STAT1
960         INC   H
970 STAT1   LD    A,(HL)
980         POP   HL
990         RET
```

Эта маленькая, но полезная процедура поможет программисту узнать, в каком состоянии находится тот или иной канал. Например, чтобы найти, какой из них свободен.

Следующая процедура - MUTE:

```
1000 MUTE   DI
1010        PUSH  HL
1020        PUSH  AF
1030        LD    HL,MUTS
1040        CP    A,3
1050        JR    Z,MUT2
1060        ADD   A,L
1070        LD    L,A
1080        JR    NC,MUT1
1090        INC   H
1100 MUT1   LD    (HL),B
1110        POP   AF
1120        POP   HL
1130        EI
1140        RET
1150 MUT2   LD    (HL),B
1160        INC   HL
1170        LD    (HL),B
1180        INC   HL
1190        JR    MUT1
```

Эта процедура понадобится, например, для временной остановки работы одного из каналов. Теперь - процедуры инициализации каналов:

```
1200 SNEWC  PUSH  IX
1210        LD    IX,CHAN_C
1220        PUSH  BC
1230        LD    B,2
1240        JR    SNEW1
1250 SNEWB  PUSH  IX
1260        LD    IX,CHAN_B
1270        PUSH  BC
1280        LD    B,1
1290        JR    SNEW1
1300 SNEWA  PUSH  IX
1310        LD    IX,CHAN_A
1320        PUSH  BC
1330        LD    B,0
```

```
1340 SNEW1  DI
1350        PUSH  BC
1360        PUSH  DE
1370        PUSH  HL
1380        PUSH  IX
1390        POP   HL
1400        PUSH  HL
1410        POP   DE
1420        INC   DE
1430        LD    BC,20
1440        LD    (HL),B
1450        LDIR
1460        POP   HL
1470        PUSH  HL
1480        LD    (IX+20),A
1490        LD    (IX+0),L
1500        LD    (IX+1),H
1510        LD    (IX+19),0
1520        LD    E,(HL)
1530        INC   HL
1540        LD    D,(HL)
1550        INC   HL
1560        LD    (IX+2),L
1570        LD    (IX+3),H
1580        LD    (IX+16),E
1590        LD    (IX+17),D
1600        LD    HL,SVOLS+30
1610        LD    (IX+12),L
1620        LD    (IX+13),H
1630        LD    (IX+14),L
1640        LD    (IX+15),H
1650        LD    (IX+18),13  ; длительность по умолчанию = 1/4
1660        POP   HL
1670        POP   DE
1680        POP   BC
1690        LD    A,B
1700        LD    B,2
1710        CALL  MUTE
1720        LD    B,1
1730        CALL  TEMPO
1740        POP   BC
1750        POP   IX
1760        EI
1770        RET
```

И, наконец, инициализация всех трёх каналов:

```
1780 SNEW   PUSH  HL
1790        CALL  SNEWA
1800        PUSH  DE
1810        POP   HL
1820        CALL  SNEWB
1830        PUSH  BC
1840        POP   HL
1850        CALL  SNEWC
1860        POP   HL
1870        RET
```

Подпрограммы инициализации подготавливают в массивах переменных все необходимые для работы данные. Они устанавливают адреса блоков изменения частоты тона и шума в положение "не используется" (заносят в них 0). Выбирают постоянный уровень громкости (15). А также устанавливают длительность нот по умолчанию равной 1/4 секунды.

Вот подпрограмма подключения второго режима прерываний:

```
1880 SINIT  LD    A,24
1890        LD    (65535),A
1900        LD    A,195
1910        LD    (65524),A
```

```
1920         LD    HL,INTR     ; HL=aflpec обработчика
1930         LD    (65525),HL
1940         LD    HL,65024
1950         LD    DE,65025
1960         LD    BC,256
1970         LD    (HL),255
1980         LD    A,H
1990         LDIR
2000         DI
2010         LD    I,A
2020         IM    2
2030         LD    HL,MUTS     ; запрет
2040         XOR   A
2050         LD    (HL),A      ; работы
2060         INC   HL
2070         LD    (HL),A      ; всех
2080         INC   HL
2090         LD    (HL),A      ; каналов
2100         EI
2110         RET
```

А это подпрограмма, возвращающая стандартный режим прерываний и выключающая сопроцессор:

```
2120 SSTOP   DI
2130         LD    A,63
2140         LD    I,A
2150         IM    1
2160         EI
2170         LD    HL,AYREGS
2180         LD    DE,AYREGS+1
2190         LD    ВСДЗ
2200         LD    (HL),B      ; B=0
2210         LDIR              ; очистка области регистров
2220         LD    A,7
2230         DEC   B           ; B=255
2240         CALL  SETAY       ; R7=255 (выключение микшера)
2250         JP    AYOUT       ; вывод регистров в сопроцессор
```

Теперь займёмся обработчиком прерываний. Так как он должен обслуживать три канала, а массивы переменных легче всего адресовать регистром IX, можно предложить такую подпрограмму:

```
2260 INTR    PUSH  AF          ; сохранение регистров
2270         PUSH  HL
2280         PUSH  DE
2290         PUSH  BC
2300         PUSH  IX
2310         LD    IX,CHAN_A   ; подготовка регистров
2320         XOR   A
2330         CALL  DISPAT      ; сопроцессора
2340         LD    IX,CHAN_B
2350         LD    A,1         ; для всех каналов
2360         CALL  DISPAT
2370         LD    IX,CHAN_C
2380         LD    A,2
2390         CALL  DISPAT
2400         CALL  AYOUT       ; вывод регистров в сопроцессор
2410         POP   IX          ; восстановление регистров
2420         POP   BC
2430         POP   DE
2440         POP   HL
2450         POP   AF
2460         RST   56          ; вызов стандартного обработчика
2470         RET
```

Этот обработчик для каждого из каналов вызывает процедуру - диспетчер (DISPAT), занося в регистр А номер канала, а в регистр IX - адрес массива его переменных.

Роль процедуры DISPAT заключается в обработке переменных TEMPS, CURTS и MUTS для указанного канала, а также в вызове основной подпрограммы создания звука - GETSND.

Вот текст процедуры DISPAT:

```
2480 DISPAT PUSH  AF
2490        LD    E,A
2500        LD    D,0
2510        CALL  STATUS
2520        OR    A            ; канал остановлен ?
2530        JR    NZ,DISP1
2540        POP   AF
2550        LD    B,A
2560        JR    DISP3
2570 DISP1  LD    HL,CURTS
2580        ADD   HL,DE
2590        LD    A,(HL)
2600        OR    A
2610        JR    Z,DISP2
2620        DEC   (HL)         ; уменьшение счетчика темпа
2630        POP   AF
2640        RET
2650 DISP2  DEC   HL           ; обновление
2660        DEC   HL
2670        DEC   HL           ; счетчика
2680        LD    A,(HL)
2690        INC   HL           ; темпа
2700        INC   HL
2710        INC   HL
2720        LD    (HL),A
2730        POP   AF
2740        CALL  GETSND       ; вызов основной процедуры
2750        LD    B,A
2760        CALL  STATUS
2770        CP    1            ; надо "заглушать" ?
2780        RET   NZ
2790 DISP3  LD    C,B          ; "заглушение"
2800        LD    HL,AYREGS+6
2810        LD    A,9          ; канала
2820 DISP4  SLA   A
2830        DJNZ  DISP4
2840        OR    (HL)
2850        LD    (HL),A
2860        INC   HL
2870        ADD   HL,BC
2880        LD    (HL),0
2890        RET
```

Итак, все сервисные процедуры приведены. Осталась только одна - GETSND:

```
2900 GETSND ...
```

Вот её-то написание я и предлагаю Вам. Но не пугайтесь - я все подробно объясню.

Скорее всего, процедура GETSND будет достаточно велика. Может быть даже больше всех приведенных процедур вместе взятых. Но ничего трудного в ней нет, а её объём обусловлен достаточно сложным форматом данных.

Задача процедуры GETSND сводится к формированию в определенных ячейках области AYREGS данных одного из каналов для последующего копирования их в регистры сопроцессора.

В качестве параметров этой процедуре передаётся номер канала в регистре A (чтобы она знала в каких ячейках размещать данные для частоты, громкости и т.п.) и адрес массива переменных в регистре IX. Обратите внимание, что она обязана сохранить значение регистра A! Возможно, Вам придется даже завести дополнительную переменную для его хранения.

Итак, порядок действий в процедуре GETSND следующий:

1. Проверить не равен ли нулю счётчик длительности (IX+19).
2. Если нет, то продолжить воспроизведение текущей ноты.
3. Если равен, то выбрать новую ноту и начать её воспроизведение.

В понятие "продолжить воспроизведение текущей ноты" входит следующее:

1. Уменьшить счётчик длительности.
2. Если адрес одного из дополнительных блоков равен 0, то пункты 3…5 для этого блока выполнять не надо.
3. Выбрать очередные значения из блоков изменения частот и громкости, используя переменные IX+6/IX+7, IX+10/IX+11 и IX+14/IX+15.
4. В соответствии с выбранными значениями и номером канала обновить с помощью процедур GETAY и SETAY область AYREGS (обратите внимание, что смещения частот могут быть и отрицательными).
5. Обновить переменные по адресам IX+6/IX+7, IX+10/IX+11 и IX+14/IX+15 в соответствии с пунктом 3.

"Начать воспроизведение ноты" включает в себя только один пункт:

1. Переписать переменные по адресам IX+4/IX+5, IX+8/IX+9, IX+12/IX+13 и IX+18 в их счётчики - дубли (IX+6/IX+7, IX+10/IX+11, IX+14/IX+15 и IX+19).

А вот пункт "выбрать новую ноту" - посложнее:

1. Выбрать очередной байт из текущего паттерна (адрес - IX+16/IX+17).
2. Если он не равен 0…100, 130, 131 и 132 - обработать как соответствующий управляющий код и перейти к пункту 1.
3. Соответственно с байтом или его параметрами и номером канала обновить область AYREGS и переменную 1Х+16ЯХ+17.

Теперь об обработке управляющих кодов:

Код 128:

1. Выбрать адрес следующего паттерна из основного блока (адрес в основном блоке содержится в IX+2/IX+3).
2. Обновить переменную IX+2/IX+3.
3. Если адрес паттерна равен нулю, скопировать переменную IX+2/IX+3 в IX+0/IX+1 и перейти к пункту 1.
4. Если адрес равен 65535, скопировать переменную IX+0/IX+1 в IX+2/IX+3 и уменьшить счётчик повторений (IX+20). Если счётчик повторений равен нулю, установить канал в состояние "остановлен" с помощью процедуры MUTE. Перейти к пункту 1.
5. Записать выбранный адрес в IX+16/IX+17 и "выбрать новую ноту".

Код 129:

1. Взять байт, следующий за этим кодом и занести в IX+18. "Выбрать новую ноту".

Код 133 (134):

1. Взять адрес, следующий за этим кодом и поместить его в IX+4/IX+5 (IX+8/IX+9). "Выбрать новую ноту".

Код 135:

1. Взять адрес, следующий за этим кодом.
2. Если он меньше 16, вычислить соответствующий адрес в таблице SVOLS.
3. Поместить полученный адрес в IX+12/IX+13.
4. "Выбрать новую ноту".

Код 136:

1. Установить в регистре громкости заданного канала значение 16.
2. Записать в IX+14/IX+15 ноль.
3. Взять байт, следующий за этим кодом.
4. Вычислить по таблице ENVS значение R13 и обновить область AYREGS.
5. Взять два байта, следующие за кодом формы и занести их в ячейки 11 и 12 области AYREGS.
6. "Выбрать новую ноту".

Теперь о том, как можно рассчитать номера регистров для заданного канала. Чтобы рассчитать номер регистра частоты достаточно номер канала умножить на 2 (ADD A,A). Полученное число будет номером регистра младшего байта частоты. Чтобы получить номер регистра старшего байта частоты, полученное значение надо увеличить на 1 (INC A).

Чтобы вычислить номер регистра громкости надо к номеру канала прибавить 8 (ADD A,8).

Если Вы напишите процедуру GETSND, в Ваших руках окажется довольно мощная программа, пригодная для написания как музыки, так и эффектов.

И в конце описания этой программы - советы по составлению для неё блоков данных.

Чтобы ноту повысить или понизить на октаву необходимо её значение соответственно увеличить или уменьшить на 12. Ноте ДО первой октавы соответствует число 39.

Значения длительностей Вы можете взять из таблицы 11.

Нота		Значение
Одна шестнадцатая	𝅘𝅥𝅯	3
Одна шестнадцатая с точкой	𝅘𝅥𝅯.	5
Одна восьмая	𝅘𝅥𝅮	6
Одна восьмая с точкой	𝅘𝅥𝅮.	9
Одна четвертая	𝅘𝅥	13
Одна четвертая с точкой	𝅘𝅥.	19
Одна вторая	𝅗𝅥	25
Одна вторая с точкой	𝅗𝅥.	38
Целая	𝅝	50

Табл. 11. Длительности нот.

Теперь несколько советов по программированию сопроцессора.

Для формирования новых тембров можно использовать генератор огибающей, настроенный на периодически изменяющуюся громкость и большую частоту. Особенно хороших результатов можно добиться, настроив его на частоту, кратную частоте основного сигнала.

Для заглушения музыкального сопроцессора довольно часто используется запрещение всех функций микшера (вывод байта 255 в R7), но такой способ не очень надежен. Если генератор огибающей настроен на периодически изменяющуюся громкость, то этот трюк не пройдет: останется слышным щелканье. Для полного заглушения сопроцессора могу посоветовать такую подпрограмму:

```
10          LD    HL,DATA     ; HL=адрес данных
20          LD    E,10        ; E=номер первого регистра
30 LOOP     LD    BC,65533    ; BC=адрес порта регистров
40          OUT   (C),E       ; вывод E в порт BC
50          LD    A,(HL)      ; A=значение очередного регистра
60          LD    B,191       ; BC=адрес порта данных
70          OUT   (C),A       ; вывод A в порт BC
80          INC   HL          ; HL=HL+1
90          DEC   E           ; E=E-1
100         LD    A,E         ; E=
110         CP    6           ; 6 ?
120         JR    NZ,LOOP     ; если нет, то цикл
130         RET               ; возврат
140 DATA    DEFB  0,0,0,255   ; данные для регистров AY
```

Во многих случаях требуется определить присутствует ли сопроцессор в данном компьютере. Некоторые делают это проверяя тип компьютера (48К/128К), но этот способ не совсем справедлив, ведь AY может стоять и на старом добром Спектруме. Вот подпрограмма, определяющая присутствие сопроцессора более достоверно:

```
10          LD    BC,65533    ; BC=адрес порта регистров
20          XOR   A           ; A=0
30          OUT   (C),A       ; выбор регистра 0
40          LD    B,191       ; BC=адрес порта данных
50          OUT   (C),A       ; вывод 0 в выбранный регистр
60          LD    B,255       ; BC=адрес порта регистров
70          IN    A,(C)       ; ввод значения из выбранного регистра
80          OR    A           ; A=0?
90          RET               ; возврат
```

Если после вызова этой подпрограммы флаг Z сброшен, то сопроцессор присутствует. В противном случае - нет.

И напоследок хочу рассказать об одном довольно интересном приёме. Он очень часто используется во многих программах. Читая данные из регистров громкости и преобразовывая их соответствующим образом в графическую информацию одновременно с воспроизведением

музыки, можно сделать цветомузыку или пиковые индикаторы уровня сигнала.

7. Обзор программного обеспечения.

Для ZX-Spectrum создано довольно много программ, позволяющих создавать различные звуки, но, к сожалению, многие из них не производят положительного впечатления.

Первую группу программ, которую я опишу, можно окрестить редакторами звуковых эффектов. Они весьма немногочисленны.

Программу SPECSOUND фирмы OZ Software отличает очень неудобный пользовательский интерфейс и возможность редактирования всего одного эффекта.

Программа SOUND FX фирмы Dk'tronics довольно неплохое произведение. Она позволяет создавать как эффекты с чистым тоном, так и шумовые эффекты, но имеет один существенный недостаток: эффекты выбирает она сама, случайным образом, редактировать их в прямом смысле этого слова невозможно.

Существует также пакет DZWIEKI, который мне, к сожалению, достать не удалось. Некоторые утверждают, что это первая версия программы SPECSOUND.

Программа SUPER SOUND, написанная автором этой книги (её подробное описание Вы можете найти в следующей главе) позволяет редактировать десять различных эффектов. Среди них есть как тоновые, так и шумовые эффекты. Кроме того, она позволяет вводить звуковые фрагменты с магнитофонного входа Спектрума и воспроизводить в различных режимах.

Все вышеперечисленные редакторы эффектов позволяют записать отредактированные эффекты на магнитный носитель в виде подпрограмм в кодах, которые можно использовать в своих программах.

Программ, позволяющих создавать звуковые эффекты для музыкального сопроцессора, мне вообще не удалось обнаружить. Но это будет исправлено с выходом следующей версии SUPER SOUND, что ожидается довольно скоро.

Следующая группа программ позволяет записывать и воспроизводить звуковые фрагменты через магнитофонный вход ZX-Spectrum. Типичным представителем является SPEAK EASY, а лидером - VOICE MANIPULATOR, написанный Джулианом Спенсером (Julian Spencer) в 1991 году.

У программ такого рода существует два недостатка: качество воспроизводимого ими звука посредственное и длина его ограничена несколькими десятками секунд (см. главу 3.5).

Довольно интересны программы, позволяющие Вашему Спектруму говорить человеческим языком. Правда, произношение у них сильно страдает и говорят они с акцентом. Пожалуй, самой шепелявой из них является BASZED. Понять, что она говорит можно лишь читая произносимую фразу на экране. В народе её даже окрестили "базетом". Несколько более понятно говорит программа FONGEN (генератор фонем). Кроме того, её можно заставить говорить по-русски.

На этом список не заканчивается. Существуют и другие такие программы: LMOWA, TOKER и т.д.

Существует также, по-моему, единственный в своем роде цифровой частотомер - DIGITAL FREQUENCER (DFR). Хотя идея и неплохая, оформлена она страшненько.

Несколько программ позволяют посмотреть диаграммы спектра звукового сигнала, подаваемого на магнитофонный вход ZX-Spectram. Например, TAPER и TAPE DIAGNOSTICS могут разложить звук на частоты до 4 кГц. Программа LIGHT SHOW, написанная Жигой Турком (Ziga Turk) в 1984 году, позволяет анализировать звук с частотой до 16 кГц. Кроме того, в ней имеется цветомузыкальный режим, создающий на экране цветовые эффекты в такт музыке, подаваемой на магнитофонный вход.

Следующая группа программ - музыкальные редакторы. Их существует довольно много. Простейшие из них, например, Menzer Synthetizer или Organ, делают из ZX-Spectram что-то вроде детской пищалки, в роли, клавиатуры которой выступают клавиши Спектрума. Более сложные программы, такие как Spectrum Musicmaker или Music Typewriter, позволяют сохранить мелодию на ленте, с последующей загрузкой и воспроизведением.

Программа A.E.Drums фирмы Einstein Software специализирована на ударных инструментах. В неё заложено десять различных ударников и столько же готовых ритмов.

Одним из самых мощных музыкальных редакторов является WHAM THE MUSIC BOX,

созданный фирмой Mark Soft в 1985 году. Он позволяет писать двухголосные мелодии с ударными инструментами. Его подробное описание Вы найдете в одной из следующих глав.

Имеется подозрение, что существуют два великолепных музыкальных редактора, превосходящих по своим возможностям WHAM THE MUSIC BOX, но недоступных рядовому пользователю. Результат работы одного из них Вы можете услышать в таких программах, как STARWARS, GOLDEN AXE, GRAND PRIX, OPERATION WOLF. Музыку из другого мне удалось обнаружить только в программах фирмы Code Masters (TWIN TURBO V8, RALLY CROSS), что наводит на некоторые мысли.

Немало существует музыкальных редакторов и для сопроцессора. Два наиболее известных - это WHAM THE MUSIC BOX 128K и ASC Sound Master (ASM). Первый написан той же фирмой Mark Soft, а второй - Андреем Сендетским из Днепропетровска (фирма Andrew Strikes Code). Хотя, WHAM 128K один из лучших музыкальных редакторов для сопроцессора, до ASM ему еще далеко. Описание ASM Вы можете найти в электронном журнале SPECTROFON № 2.

Пожалуй, последней группой звуковых программ являются музыкальные демонстрационные программы для сопроцессора. Их можно разделить на две подгруппы. Первая - музыкальные сборники, а вторая - динамические шоу.

Типичными представителями первой подгруппы являются, например, MANHATTAN или TOP 128. Эти программы не очень интересны, но являются крупными библиотеками мелодий.

Из второй подгруппы можно выделить две наиболее красивые программы. Это SHOCK и THE LYRA II польской фирмы ETHANOL SOFT INC. В каждой из них содержится по несколько частей (8 и 9 соответственно) с прекрасной графикой и музыкой.

Данный обзор не претендует на полноту, но я постарался охватить максимальное количество видов программ.

7.1. Редактор звуковых эффектов SUPER SOUND.

Программа SUPER SOUND v2.2 позволяет выбрать и настроить под собственные нужды один из одиннадцати различных звуковых эффектов с последующей выгрузкой на диск или ленту, в зависимости от версии. Все файлы, выгруженные SUPER SOUND'om являются полностью перемещаемыми, законченными подпрограммами в машинных кодах, и могут вызываться как из бейсика, так и на ассемблере.

SUPER SOUND работает на всех типах Спектрум-совместимых компьютеров и в музыкальном сопроцессоре (AY-3-8912) не нуждается.

Программа управляется с помощью курсора и меню. Выбор функций осуществляется наведением курсора на соответствующий пункт в текущем окне и нажатием кнопки "огонь". Для управления курсором можно использовать Кемпстон джойстик, Синклер джойстик или клавиатуру (клавиши O, P, Q, A и SPACE - влево, вправо, вверх, вниз и огонь соответственно). Все устройства управления активны одновременно.

Главное меню содержит двенадцать пунктов. Первые одиннадцать - различные эффекты, а последний (Info) - информация о программе, дате её создания и авторе.

7.1.1. Эффекты.

Опишу все эффекты по очереди. При выборе любого из них перед Вами раскроется окно с текущими параметрами данного эффекта и несколькими служебными функциями. В каждом из одиннадцати окон существуют три общие функции: Main Menu - возврат в главное меню, Play - воспроизведение эффекта и Save - запись настроенного эффекта на магнитный носитель. Остальные функции изменяются от эффекта к эффекту.

Input Sound - ввод звука. Этот пункт главного меню позволяет вводить в память звуковые фрагменты, подаваемые на магнитофонный вход компьютера и воспроизводить их с различной скоростью. В окне этого эффекта кроме трех стандартных функций существует несколько собственных: Input Speed:XX -скорость ввода. Чем меньше данный параметр, тем лучше качество вводимого звука, тем больше он занимает памяти и тем в меньшее число раз его можно ускорить. Если Вам необходимо максимальное качество, то скорость ввода должна быть равна 0, но тогда то, что Вы ввели можно будет воспроизводить либо с реальной скоростью, либо замедлить. Play

Speed:XX - скорость воспроизведения. Если параметр этого пункта равен параметру Input Speed, то звуковой фрагмент воспроизводится с реальной скоростью, если он меньше, то звук ускоряется, если же он больше - звук замедляется. Length:XXXXX - длина звукового фрагмента. Reverse - обратное воспроизведение. Этот пункт позволяет "переворачивать" Ваш звуковой фрагмент с ног на голову, то есть воспроизводить его с конца на начало. Когда напротив этого пункта установлена галочка, данный режим включен. Input - ввод звука. При выборе этого пункта сразу же начинается запись звука в память. Поэтому, перед тем, как это сделать, проверьте Ваш магнитофон и шнуры, соединяющие его с компьютером.

Double Beep - двойной BEEP. Название этого пункта говорит само за себя - это аналог оператора бейсика BEEP, но воспроизводящий две частоты одновременно. Frequency 1:XXX и Frequency 2:XXX - соответственно первая и вторая частоты. Duration:XXX - длительность эффекта. Пункты Main Menu, Play и Save описаны выше.

Exploding I - имитатор взрыва 1. Этот эффект представляет собой шум с плавно изменяющейся частотой. Кроме трёх стандартных функций в этом окне вы встретите следующие: Frequency:XXX - начальная частота эффекта. Duration:XXX - длительность звучания одной ступени частоты. Length:XXX -длина эффекта. Group:XX - этот параметр влияет на звук незначительно. Его значение определяет адрес ПЗУ, из которого берутся данные для генерации шума. Increase - если Вы выберете этот пункт, то он сменит свое название на Decrease (уменьшение). Если же Вам этого покажется мало, и Вы опять остановите свой выбор на этом пункте, то он вновь превратится в Increase (увеличение). Этот пункт определяет направление смещения частоты шума.

Exploding 2 - имитатор взрыва 2. Этот эффект похож на предыдущий, но моменты изменения частоты заметны гораздо больше. При редактировании этого эффекта у Вас может вызвать затруднение только пункт Length:XXX - это длина всего эффекта. Остальные (Frequency:XXX, Increase, Main Menu, Play и Save) полностью аналогичны таким же, описанным выше.

Volume FX - громкостный эффект. Это одна из основных достопримечательностей программы SUPER SOUND. Данный пункт главного меню позволяет создавать эффекты с различной громкостью и даже с изменением её в течение воспроизведения. Пункты Frequency: XXX, Duration:XXX и Length:XXX полностью аналогичны таким же в Exploding 1. Пункт Volume:XX устанавливает начальную громкость эффекта.

Следующие три пункта следует объяснить поподробнее. Это: FQ:Increase, DR: Increase и VL:Decrease. Эти пункты определяют изменение частоты, длительности и громкости, соответственно. Они могут принимать значения Increase, Decrease и No change (увеличение, уменьшение и неизменение). Обратите внимание, что из-за организации этого эффекта, если Вы установите изменение громкости, то это необходимо компенсировать изменением частоты в противоположную сторону. Иначе эта частота будет смещаться. Последние три пункта этого эффекта стандартны (см. выше).

Flowing 1 - плавный эффект 1. Этот эффект представляет собой тон с плавно изменяющейся частотой. Все пункты в этом окне аналогичны таким же в Exploding 1, поэтому приведу только их список, без описания: Frequency:XXX, Duration:XXX, Length:XXX, Increase (Decrease), Main Menu, Play и Save.

Flowing 2 - плавный эффект 2. Этот эффект более изощрён, чем предыдущий. Частота тона в конкретный момент времени складывается из двух составляющих, которые могут изменяться независимо друг от друга. В меню эти составляющие названы Frequency I :XXX и Frequency 2:XXX, а способы их изменения Fl:<mode> и F2:<mode>, где <mode> может принимать значения Increase, Decrease и No change (см. выше). Пункт Duration:XXX полностью похож на такой же в Exploding I, а Main Menu, Play и Save - стандартны.

Cycle 1 - цикловой эффект 1. Этот и следующий эффекты основаны на подпрограмме BEEP из ПЗУ ZX-Spectram. Cycle 1 очень похож по свойствам на Flowing 1, но тембр его звучания значительно отличается. Пункты Frequency:XXXX, Duration:XXXX, Increase (Decrease), Main Menu, Play и Save аналогичны таким же во Flowing 1. Пункт Quantity:XXX - определяет количество проходов цикла. Он похож на параметр Length:XXX предыдущих эффектов.

Последний неизвестный пункт - Step:XXX. Он определяет шаг изменения частоты.

Cycle 2 - цикловой эффект 2. От Cycle 1 этот эффект отличается только тем, что изменение частоты ограничено и при переходе определенного порога её значение становится близким к начальному, что позволяет создавать довольно интересные варианты. Все пункты этого эффекта полностью аналогичны таким же в Cycle 1.

Noise I - шум 1. Этот эффект представляет собой обыкновенный шум. Пункты Frequency:XXX, Length:XXXX, Group:XX, Main Menu, Play и Save аналогичны таким же в Exploding 1. Bounds - скачки. Если этот пункт отмечен галочкой, то шум становится как бы прыгающим.

Noise 2 - шум 2. Это шумовой аналог Flowing 2, поэтому описания всех пунктов этого эффекта Вы можете найти в Flowing 2. Исключение составляет только Group:XX, его описание находится в Exploding 1.

7.1.2. Использование эффектов.

Теперь немного об использовании выгруженных эффектов. Любой из них можно загрузить в память, набрав с клавиатуры

`LOAD "name" CODE addr` для ленты и

`RANDOMIZE USR 15619: REM: LOAD "name" CODE addr` для диска

- где name - имя файла на ленте или диске, а addr - адрес в который необходимо загрузить эффект. Чтобы запустить загруженный эффект, введите

`RANDOMIZE USR addr.`

Обратите внимание, что файлы, записанные из окна Input Sound, как правило, очень велики по размерам, поэтому, при их загрузке следите, чтобы они не перекрыли системные области ОЗУ.

7.1.3. Версии.

Существует несколько различных версий SUPER SOUNDa:

1.0 - Это пробная версия. Она страдает практически всеми недостатками, которые можно представить и лучше её никому не видеть.

2.0 - Программа значительно улучшена. Она содержит динамическую заставку. Эта версия послужила основой для всех последующих.

2.1 - Удалена заставка, добавлен контроль диапазона в Cycle 1 и новый эффект - Volume FX.

2.2 - Улучшен дизайн, исправлены мелкие ошибки в подпрограммах воспроизведения.

Планируется выход версии 3.0, которая будет на порядок превосходить своих предшественниц. Предполагается расширить возможности Information, Input Sound и Double Beep, добавить компилятор и возможность создания эффектов для сопроцессора, несколько новых эффектов и полезных функций.

Версия 1.0 существует только в кассетном исполнении. Версии 2.0 и 2.1 - как в кассетном, так и в дисковом. Версия 2.2 - только в дисковом. Версия 3.0 будет рассчитана на Spectrum 128K и на компьютерах с меньшим объёмом памяти некоторые функции работать не будут.

7.2. Музыкальный редактор Wham the Music Box.

Wham the Music Box ("музыкальная шкатулка") - пожалуй, самый богатый по своим возможностям музыкальный редактор для ZX-Spectram. С его помощью Вы сможете создавать двухголосые мелодии с использованием барабана и шумовых эффектов. Созданный музыкальный фрагмент можно сохранить на ленте или дискете и, при необходимости, загрузить его снова и внести изменения. Большим достоинством редактора является то, что мелодии, написанные с его помощью, можно использовать для оформления других программ.

После загрузки Wham'a звучит мелодия, а на экране появляется главное меню (MAIN MENU). Для выбора какого-либо режима нажмите соответствующую цифровую клавишу:

1. LOAD TUNE - загрузка мелодии
2. SAVE TUNE - сохранение мелодии
3. HEAR TUNE - прослушивание мелодии
4. WHAMPILER - компиляция

5. SET TEMPO - установка темпа
6. EDIT MODE - режим редактирования
7. HELP PAGE - подсказка

7.2.1. LOAD TUNE - загрузка мелодии.

Этот режим позволяет загрузить файл, содержащий ранее созданную мелодию, в редактор для дальнейшей работы с ним. После перехода в режим LOAD TUNE (клавиша 1) программа спросит, с какого устройства Вы хотите загружать файл:

```
SELECT PERIFERAL DEVICE
TAPE MEMORY DRIVE (SPACE=EXIT)
```

Если Вы решили отказаться от своей затеи, то нажмите пробел. В противном случае - выбирайте устройство с помощью одной из клавиш: T - кассета, M - оперативная память, D - микродрайв.

При загрузке с кассеты на запрос "FILENAME ?" введите имя файла, в котором хранится мелодия.

Вместе с Wham'ом на кассете (дискете) поставляется файл EXAMPLE, в котором содержится весёлая мелодия. Попробуйте его загрузить и прослушать (см. дальше).

Так как в России микродрайв не прижился, а вместо него прижился дисковод, возникает законное желание заменить надпись DRIVE на DISK, со всеми вытекающими отсюда последствиями. Это было довольно успешно сделано несколькими российскими программистами (существует также версия программы на русском языке). Самый распространенный дисковый вариант перед запросом "FILENAME ?" (см. выше) спрашивает не распечатать ли ему каталог диска "A" (Да/Нет):

```
PRINT CATALOGUE 'A:' (Y/N) ?
```

А затем загрузка идёт стандартным образом.

Если Вы обратитесь к оперативной памяти (клавиша M), то на экране появится список из шести мелодий:

1. FREEDOM
2. TROPICANA
3. YOUNG GUNS
4. WHISPER
5. BAD BOYS
6.

Первые пять из них созданы авторами программы, а шестая зарезервирована для Ваших творений. Для загрузки любой из них достаточно нажать соответствующую цифровую клавишу.

По окончании загрузки программа автоматически переходит в режим MAIN MENU.

7.2.2. SAVE TUNE - сохранение мелодии.

Благодаря этому режиму Ваш труд не пропадёт даром, когда Вам придётся надолго оторваться от составления мелодии, то есть созданный музыкальный фрагмент можно будет сохранить на ленте или диске для дальнейшего усовершенствования.

Войти в режим SAVE TUNE можно из главного меню, нажав клавишу 2, после чего появится уже знакомый вопрос об устройстве, на которое Вы желаете записать свое произведение:

```
SELECT PERIFERAL DEVICE
TAPE MEMORY DRIVE (SPACE=EXIT)
```

Как и раньше, нажав пробел, Вы вернетесь в главное меню, а выбрав устройство: T - кассета, M - память и D - микродрайв (дискета), получите запрос "FILENAME ?", на который надо ввести имя файла, в котором будет сохранена мелодия.

Если Вы решили записать свой музыкальный фрагмент в оперативную память, перед Вами появится список из шести мелодий. Ваше произведение можно записать на место любой из них, но храниться оно будет только до выключения питания или нажатия кнопки сброса.

В случае записи на кассету, после её завершения появится вопрос: VERIFY (Y/N). Если Вы уверены в качестве записи, то можете жать клавишу "N", в противном случае нажмите "Y",

перемотайте ленту на начало только что записанного файла и включите воспроизведение для его проверки.

По окончании записи программа автоматически переходит в режим MAIN MENU.

7.2.3. EDIT MODE - режим редактирования.

Режим редактирования - это основной инструмент, с помощью которого Вы сможете реализовать свои музыкальные идеи.

Войти в этот режим можно из главного меню, нажав клавишу 6. При этом на экране появятся изображения двух нотных станов (для басового и скрипичного ключей), а также указатель октавы OCTAVE, указатель текущего канала (голоса) CHANNEL 1 (2) и счётчики шагов для каждого из каналов COUNTERS (CHN1, CHN2).

Для выбора функций редактора используются два верхних ряда клавиш, а нижние два - заменяют собой клавиатуру фортепиано. Значения этих клавиш Вы можете увидеть на рис. 5. Ряд от Caps Shift до Space играет роль белых клавиш, а от A до L - черных.

1	2	3	4	5	6	7	8	9	0
ВЫБОР ОКТАВЫ				БОРДЮР	ГЛАВНОЕ МЕНЮ	УДАЛИТЬ МЕЛОДИЮ	ВЫБОР ЭФФЕКТОВ	ПРОКРУТ-КА НАЗАД	ШАГ НАЗАД

Q	W	E	R	T	Y	U	I	O	P
ПРОИГРЫШ МЕЛОДИИ	МЕТКА ЦИКЛА	БАРАБАН	В НАЧАЛО МЕЛОДИИ	ВЫБОР КАНАЛА	ШУМОВЫЕ ЭФФЕКТЫ			ПРОКРУТ-КА ВПЕРЕД	ШАГ ВПЕРЕД

A	S	D	F	G	H	J	K	L	ENTER
ДО#	РЕ#	—	ФА#	СОЛЬ#	ЛЯ#	—	ДО#	РЕ#	ПАУЗА

CAPS SHIFT	Z	X	C	V	B	N	M	SYMBOL SHIFT	SPACE
ДО	РЕ	МИ	ФА	СОЛЬ	ЛЯ	СИ	ДО	РЕ	МИ
ТЕКУЩАЯ ОКТАВА							СЛЕДУЮЩАЯ ОКТАВА		

Рис. 5 Функции клавиш в режиме редактирования.

Итак, Вы хотите "набросать" какую-нибудь мелодию. Нажимая клавишу T, выберите голос, с которым собираетесь работать, клавишами 1…4 - октаву (1 -большая, 2 - малая, 3 - первая, 4 - вторая) и попробуйте "наиграть" мелодию на "звуковых" клавишах. При этом на нотном стане появятся изображения нот, а цветные квадратики укажут соответствующие клавиши фортепиано (для первого голоса квадратик красный, для второго - фиолетовый). Если Вы ошибочно нажали не ту клавишу, не расстраивайтесь, все поправимо: нажав клавишу 0, Вы вернетесь на один шаг назад. Если же нужно вернуться больше, чем на шаг, нажмите 9.

Проигрывание одной ноты с продвижением на шаг вперед осуществляется клавишей P. Удерживая эту клавишу Вы можете прослушать часть мелодии. С помощью клавиши O можно "перемотать" мелодию вперёд.

Вам наверняка захочется прослушать музыкальный фрагмент целиком, не выходя из режима редактирования. Нажмите R (возврат в начало мелодии), затем Q (проигрывание мелодии), - и зазвучит Ваше творение. Остановить воспроизведение можно нажатием любой клавиши.

Для написания второго голоса переключите клавишей T канал и с помощью клавиш R, O, P, 0, 9 перейдите к тому шагу, с которого Вы собираетесь его использовать.

К сожалению, Wham не даёт возможности записать мелодию нотами разной длительности. Необходимая длительность достигается либо использованием паузы (клавиша Enter), либо записью нужного количества одинаковых нот.

Ваш музыкальный фрагмент зазвучит совсем по другому, если Вы используете для его аранжировки барабан (клавиша E) и шумовые эффекты (клавиши Y, U, I). Барабан может звучать по разному, в зависимости от того, в каком канале он установлен. В первом канале это действительно барабан, а во втором он больше похож на ноту СИ контроктавы или ДО большой октавы. Шумовые эффекты можно редактировать с помощью соответствующего режима, в который можно попасть из EDIT MODE, нажав клавишу 8. После этого на экране появится меню выбора частоты (WAVEFORM) и длительности (DURATION) шумовых эффектов. Выбрать один

из предоставляемых программой 16 вариантов длительности и 8 вариантов частоты (формы колебаний) шумовых эффектов можно с помощью следующих клавиш:

5,8 - Выбор редактируемого шумового эффекта;
6 - Переход к функции выбора частоты;
7 - Переход к функции выбора длительности;
0 - Изменение эффекта.

Получающиеся в процессе редактирования шумовые эффекты можно прослушать, используя клавишу 9.

После настройки эффектов можно выйти обратно в режим редактирования мелодии (любая клавиша, кроме участвующих в настройке эффектов) и использовать их в работе с помощью клавиш Y, U и I. Хотя под шумовые эффекты отведены только эти три клавиши, Вы можете использовать гораздо большее их количество (до 128). Для этого создаются первые три эффекта и расставляются в нужных местах мелодии, затем следующие три и т. д.

Наличие барабана и шумовых эффектов в мелодии отображается на нотном стане в виде пробела, что очень неудобно. Учтите, что одновременно с барабаном и шумовыми эффектами не может звучать нота (исключение составляет только барабан, установленный во втором канале).

Если Вы хотите, чтобы мелодия повторялась с какого-либо шага, то есть звучала непрерывно, клавишей W установите на выбранном шаге метку цикла в обоих каналах. При этом компьютер спросит:

`SET LOOP HERE ? (Y/N) - установить метку здесь ? (Да/Нет)`

Расположение меток в первом и втором каналах может и не совпадать. Но для того, чтобы при циклическом проигрывании мелодии не нарушалась синхронность звучания голосов, количества шагов в первом и втором каналах должны быть кратны друг-другу. Для успешной компиляции (см. ниже) установка метки в обоих каналах обязательна.

Если Вы хотите очистить память от неудачного произведения, нажмите 7 и подтвердите запрос:

`ERASE CURRENT TUNE ? (Y/N) - удалить мелодию ? (Да/Нет)`

С помощью клавиши 5 можно изменять цвет бордюра. А чтобы вернуться в главное меню, нажмите клавишу 6.

7.2.4. HELP PAGE - подсказка.

В режиме редактирования Вам понадобится использовать немалое количество клавиш, сразу запомнить их функциональное назначение не так просто. Поэтому в качестве помощи Вам предлагается страница подсказки - HELP PAGE, которую можно вызвать из главного меню, нажав клавишу 7.

На экране появится список клавиш, используемых при редактировании:

KEYS	FUNCTION	
1-4	SELECT OCTAVE	выбор октавы
5	CHANGE BORDER	изменение цвета бордюра
6	MAIN MENU	выход в главное меню
7	ERASE TUNE	удаление мелодии
8	NOISE MIXING	редактирование шумовых эффектов
9	REPEAT BACKSTEP	прокрутка назад
O	SINGLE BACKSTEP	возврат на один шаг
Q	REPLAY TUNE	проигрывание мелодии
E	DRUM EFFECT	барабан
T	CHANGE CHANNEL	выбор канала
Y,U&I	NOISE EFFECTS	шумовые эффекты
O	FAST FORWARD	прокрутка вперед
P	PLAY NEXT NOTE	проигрывание следующей ноты

The lower half of the keyboard mimics a piano, enter is a rest	Нижняя половина клавиатуры имитирует фортепиано, enter - пауза

Use cursor keys to edit the white noise with 9 to hear the effect and 0 to alter it 9	Используйте курсорные клавиши для редактирования белого шума, current прослушивание, 0 - изменение

7.2.5. HEAR TUNE - прослушивание мелодии.

При прослушивании мелодии в режиме EDIT MODE, она звучит в несколько искаженном виде, так как процессор кроме воспроизведения музыки занят ещё и выводом на экран нот, обслуживанием клавиатуры и т. д. Чтобы получить представление об истинном звучании мелодии, то есть услышать, как она будет исполняться после компиляции (см. ниже), воспользуйтесь функцией HEAR TUNE (клавиша 3 главного меню).

После прослушивания, нажав любую клавишу, Вы попадаете в режим EDIT MODE.

7.2.6. SET TEMPO - установка темпа.

Если после прослушивания мелодии Вы захотите изменить темп её исполнения, воспользуйтесь режимом SET TEMPO (клавиша 5 главного меню).

Клавишей 8 можно ускорить темп исполнения мелодии, а клавишей 5 - замедлить. Изменение темпа изображается наглядно, с помощью фиолетовой полоски на экране. С уменьшением темпа, полоска тоже уменьшается и наоборот.

Нажав любую клавишу, кроме 5 и 8, Вы попадёте в режим редактирования (EDIT MODE).

7.2.7. WHAMPILER - компиляция.

Музыкальный фрагмент, сохраненный в виде редактируемого файла в режиме SAVE TUNE, можно загрузить снова (LOAD TUNE) и продолжить работу с ним. Однако, такой способ представления мелодии позволяет прослушивать её только находясь в редакторе.

Если же Вы хотите оформить музыкой программу, написанную на бейсике или ассемблере, необходимо записать мелодию в виде программы, работающей независимо от редактора. Такая подпрограмма создаётся с помощью режима компиляции WHAMPILER.

В результате компиляции на ленту или дискету записывается файл, который представляет собой программу в кодах, запускаемую отдельно от Wham'a, но, в отличие от редактируемого файла, не поддающуюся дальнейшему изменению.

Перед компиляцией с помощью функции LOAD TUNE загрузите мелодию, которую хотели-бы скомпилировать и проверьте, в обоих ли каналах выставлены метки цикла.

В качестве примера попробуем откомпилировать одну из 5 мелодий, загружаемых из памяти (см. LOAD TUNE - MEMORY), допустим, тему под номером 1 - FREEDOM. Загрузите её и, нажав клавишу 4, войдите в режим компиляции. На запрос TUNENAME ? введите имя, под которым Вы хотите сохранить откомпилированный файл, например, FREE-MUS. Далее программа попросит ввести адрес, с которого будет располагаться и запускаться откомпилированный файл:

ASSEMBLY ADDRESS ?

Укажите десятичный адрес не менее 32768, например, 60000, после чего компилятор выведет на экран следующую информацию:

```
       TUNE NAME: FREE-MUS    имя мелодии
ASSEMBLY ADDRESS: 60000       адрес
   RETURN OPTION: KEYPRESS    условие возврата
     WHITE NOISE: --NONE--    шумовые эффекты отсутствуют
CHANNEL 1 LENGTH: 313         число шагов в 1-ом канале
CHANNEL 1 LOOP  : START       цикл от начала в 1-ом канале
CHANNEL 2 LENGTH: 313         число шагов во 2-ом канале
CHANNEL 2 LOOP  : START       цикл от начала во 2-ом канале

1. KEYPRESS
2. ALWAYS
3. TUNEEND
```

```
RETURN OPTION 1, 2 OR 3 ?
```

Цифрами 1, 2 и 3 обозначены условия окончания воспроизведения мелодии. В зависимости от выбранного варианта получаются определенные модификации откомпилированного файла:

KEYPRESS - проигрывание мелодии завершается при нажатии любой клавиши;

ALWAYS - проигрывание мелодии осуществляется одновременно с работой программы (подробно этот режим описан ниже);

TUNEEND - проигрывание завершается либо по окончании мелодии, либо при нажатии любой клавиши.

Запуск компиляции произойдет сразу после выбора клавишами 1, 2 или 3 одного из перечисленных условий. Нажмите, к примеру, клавишу 1.

По окончании компиляции на экран будет выведена длина исполняемого файла в байтах и сообщение о нормальном завершении операции:

CODE LENGTH: 893

ROUTINE COMPILATION COMPLETED OK

Затем, после надписи ADJUSTMENT POKES: (настроечные POKE), появится информация, необходимая для настройки исполняемого файла уже в процессе использования его в Вашей программе (помните, что в нашем примере ASSEMBLY ADDRESS равен 60000):

REPLAY SPEED: 60035, (230 TO 255)

- этой надписью программа сообщает, что, записывая в ячейку памяти с адресом 60035 (ASSEMBLY ADDRESS + 35) число от 230 до 255, можно изменять темп исполнения мелодии;

BORDER COLOR: 60026, (0 TO 7)

- это сообщение говорит о том, что, записывая в ячейку памяти 60026 (ASSEMBLY ADDRESS + 26) число от 0 до 7, Вы можете на время исполнения мелодии устанавливать требуемый цвет бордюра (по умолчанию - фиолетовый);

TO RUN - RANDOMIZE USR 60000

- эта надпись напоминает, что для запуска мелодии из бейсика необходимо использовать команду RANDOMIZE USR 60000 (ASSEMBLY ADDRESS).

После вывода этих сообщений производится запись исполняемого файла на кассету или диск, в зависимости от версии.

Теперь можно смело жать кнопку сброса и загружать в память, полученную после компиляции программу в кодах. Если исполняемый файл сохранен на ленте, то для его загрузки и запуска напишите следующую программу на бейсике:

```
10 LOAD "FREE-MUS" CODE
20 POKE 60026,7: REM цвет бордюра - белый
30 RANDOMIZE USR 60000
```

Запустите её и загрузите мелодию, а затем Вы услышите скомпилированную мелодию, которая будет звучать, пока Вы не нажмете какую-нибудь клавишу. Поэкспериментируйте с темпом исполнения произведения, занося с помощью оператора POKE различные числа в пределах от 230 до 255 в ячейку 60035.

Для дисковой версии первую строку программы нужно заменить на следующую:

```
10 RANDOMIZE USR 15619: REM: LOAD "FREE-MUS" CODE
```

Теперь Вы сможете откомпилировать любую мелодию (в том числе и свою собственную), выполняя аналогичные действия.

Процесс компиляции может проходить несколько иначе. Например, если Вы решили сделать повторяющуюся басовую партию и зациклили её гораздо раньше основной мелодии (проследите, чтобы число шагов в обоих каналах было кратно друг-другу). В этом случае программа исключает возможность компиляции в режиме TUNEPEND и устанавливает режим KEYPRESS, а вместо возможности выбора спрашивает:

CHANGE RETURN OPTION TO ALWAYS ?

сменить режим компиляции на ALWAYS ?

Если Вы нажмете клавишу Y, режим будет сменен. В противном случае компиляция продолжится в режиме KEYPRESS.

Учтите, что компилятор выдает число шагов в канале вместе с меткой цикла. Поэтому Вы можете обнаружить, что, например, 65 кратно 9 (на самом деле - 64 и 8).

Если в Вашей мелодии число шагов в обоих голосах не кратно друг-другу, то компиляция прервется следующим сообщением:

WARNING	**ПРЕДУПРЕЖДЕНИЕ**
End markers mismatch	**Не совпадают метки**
цикла. may cause distorted tune	**Может исказиться мелодия.**
Continue ? (Y/N)	**Продолжить ? (Да/Нет)**

Если Вы нажмете Y, то компиляция продолжится как если бы число шагов было кратно друг-другу, но за то, что получится, никто ответственности не несёт. В противном случае компиляция прервётся и Вы окажетесь в главном меню.

Если же Вы вообще забыли поставить метку цикла хотя бы в одном из каналов, то Вашему взору предстанет следующая картина:

```
     TUNE NAME: XXXXXXXXXX
ASSEMBLY ADDRESS: XXXXX
   RETURN OPTION: KEYPRESS
     WHITE NOISE: XXXXXXXX
CHANNEL 1 LENGTH: NOT PRESENT
число шагов в первом канале не определено
CHANNEL 1 LOOP  : START
CHANNEL 2 LENGTH: NOT PRESENT
число шагов во втором канале не определено
CHANNEL 2 LOOP  : START
```

Tune compilation abandoned.	Компиляция прервана.
No end markers defined.	Не установлены метки цикла.
Use the W key to place the end	Используя клавишу W установите
marker for the current channel.	метку в каждом канале.

Если машина обругала Вас отнеситесь к этому спокойно. Нажав клавишу Enter, выйдите из режима компиляции, устраните замеченные недостатки и, будьте уверены, что со второй, в крайнем случае с третьей попытки мелодия будет откомпилирована, и Вы сможете украсить ею свою программу. Скорее всего, Вы захотите, чтобы Ваша программа работала одновременно с проигрыванием мелодии. Подобного эффекта можно добиться, если при компиляции использовать режим ALWAYS. Файл, скомпилированный в этом режиме вызывается по двум адресам: имеет две точки входа. Первая служит для инициализации мелодии и проигрывания первой ноты и расположена по тому же адресу, что и в других режимах (ASSEMBLY ADDRESS), а вторая - проигрывания всех последующих нот (расположена по адресу ASSEMBLY ADDRESS + 12).

Таким образом, если в приведённом выше примере компиляции использовать режим ALWAYS, то для инициализации мелодии и проигрывания первой ноты нужно выполнить оператор

RANDOMIZE USR 60000,

а для проигрывания каждой последующей ноты - оператор

RANDOMIZE USR 60012.

Вот пример программы, демонстрирующей возможности режима ALWAYS:

```
10 PAPER 0: BORDER 0: INK 7: BRIGHT 1: CLS
20 LOAD "FREE-MUS" CODE: CLS: OVER 1
30 LET A$="ALWAYS-mode demonstration!"
40 RANDOMIZE USR 60000
50 PRINT AT 10,3;
60 FOR A= 1 TO LEN A$: RANDOMIZE USR 60012
70 PRINT INK RND*6+1; A$(A);
80 NEXT A: GOTO 50
```

Аналогичным образом Вы можете оформлять и программы в машинных кодах. Учтите, что время выполнения Вашей программы в промежутках между нотами не должно быть очень большим, иначе теряется качество мелодии. Кроме того, оно должно быть примерно одинаковым между двумя любыми нотами.

7.2.8. Советы.

1. Если программа вышла в бейсик с сообщением об ошибке (например, при загрузке с магнитофона или если Вы совершенно случайно нажали клавишу BREAK), то для возвращения в главное меню, введите команду RUN.
2. Если Вы решили создать более менее длинную мелодию, используя оба голоса, то вводить эти голоса следует параллельно, иначе неизбежны ошибки. Кроме того, желательно время от времени прослушивать, что у Вас получилось.
3. Все сочиненные Вами мелодии стоит сохранять на одной кассете (или дискете) с помощью функции SAVE TUNE. Благодаря этому через некоторое время у Вас будет довольно большой выбор мелодий, при оформлении новой программы.

Приложение 1.

Листинги звуковых эффектов SUPER SOUNDa.

Все эффекты, приведены точно в таком виде, как в SUPER SOUND v2.2 и снабжены комментариями. Метки, начинающиеся с символов "+", "*", ":" и "?" являются специальными и отмечают изменяемые значения:

+ - однобайтовое изменяемое значение

* - два однобайтовых изменяемых значения

: - двухбайтовое изменяемое значение

? - изменяемая команда

Слово, следующее за специальным символом повторяет название пункта в программе, отвечающего за данный параметр (возможно в сокращенном виде). В случае изменяемой команды в комментарии (в скобках) указаны все возможные варианты. Для нормальной компиляции специальные метки необходимо удалить или переименовать.

Input Sound:

```
10              DI                     ; запрет прерываний
20              LD    HL,37000         ; HL=адрес сохранения звука
30 :LENGTH      LD    DE,27000         ; DE=длительность звука
40 LOOP1        LD    B,8              ; B=счетчик битов
50 LOOP2        SLA   (HL)             ; скроллинг значения в памяти
60              IN    A,(254)          ; ввести значение из порта 254
70              BIT   6,A              ; проверить бит магнитофона
80              JR    Z,+INPUT_SPD     ; если 0, то перейти на NOSIGN
90              SET   0,(HL)           ; установить бит D0 в памяти
100 +INPUT_SPD  LD    C,3              ; C=задержка
110 PAUSE       DEC   C                ; C=C-1
120             JR    NZ,PAUSE         ; если C<>0, то цикл
130             DJNZ  LOOP2            ; продолжить цикл обработки байта
140             INC   HL               ; HL=HL+1
150             DEC   DE               ; DE=DE-1
160             LD    A,D              ; DE=
170             OR    E                ; 0 ?
180             JR    NZ,LOOP1         ; если нет, то цикл
190             EI                     ; разрешение прерываний
200             RET                    ; возврат
```

Play Sound

```
10              CALL  124              ; по адресу 124 находится RET
20              DEC   SP               ; поднять указатель стека
30              DEC   SP               ; на два байта
40              POP   HL               ; снять со стека адрес строки 20
50              LD    DE,47            ; DE=число байт от строки 20 до буфера
60              ADD   HL,DE            ; HL=адрес буфера
70 :LENGTH      LD    DE,27000         ; DE=длительность звука
80 ?REVERSE     NOP                    ; резерв для REVERSE (ADD HL,DE)
90              DI                     ; запрет прерываний
100             LD    A,(23624)        ; A=
110             RRA                    ; цвет
120             RRA                    ;  бор-
130             RRA                    ;    дюра
140 LOOP3       LD    B,8              ; B=счетчик битов
150 LOOP4       AND   239              ; сброс бита D4 регистра A
160             RLC   (HL)             ; скроллинг данных через флаг CY
170             JR    NC,NOSGN         ; если флаг CY=0, то перейти на NOSGN
180             OR    16               ; установка бита D4 регистра A
190 NOSGN       OUT   (254),A          ; вывод A в порт 254
200 +PLAY_SPD   LD    C,3              ; C=задержка
210 PAUS2       DEC   C                ; C=C-1
```

```
220             JR    NZ,PAUS2      ; если C<>0, то цикл
230             DJNZ  LOOP4         ; продолжить цикл обработки байта
240 ?REVERSE    INC   HL            ; HL=HL+1 (DEC HL)
250             LD    C,A           ; сохранение A
260             DEC   DE            ; DE=DE-1
270             LD    A,D           ; DE=
280             OR    E             ; 0 ?
290             LD    A,C           ; восстановление A
300             JR    NZ,LOOP3      ; если DE<>0, то цикл
310             EI                  ; разрешение прерываний
320             RET                 ; возврат
```

Double Beep

```
10              DI                  ; запрет прерываний
20 *FRQ1,FRQ2   LD    DE,13000      ; E=частота 1, D=частота 2
30 +DURATION    LD    H,100         ; H=длительность
40              LD    A,(23624)     ; A=
50              RRA                 ; цвет
60              RRA                 ;  бор-
70              RRA                 ;    дюра
80              LD    C,A           ; сохранение A
90              EX    AF,AF'        ; смена регистров A и F на альтернативные
100             LD    A,C           ; восстановление A
110             LD    C,E           ; C=счетчик 1
120             LD    B,D           ; B=счетчик 2
130 BEEP        EX    AF,AF'        ; смена регистров A и F (смена голоса)
140             DEC   C             ; C=C-1
150             JR    NZ,CONT       ; если C<>0, топерейти на CONT
160             LD    C,E           ; восстановить счетчик 1
170             XOR   16            ; инвертировать бит D4 голоса 1
180 CONT        OUT   (254),A       ; вывести A в порт 254
190             EX    AF,AF'        ; смена регистров A и F (смена голоса)
200             DEC   B             ; B=B-1
210             JR    NZ,CONT2      ; если B<>0, то перейти на CONT2
220             LD    B,D           ; восстановить счетчик 2
230             XOR   16            ; инвертировать бит D4 голоса 2
240 CONT2       OUT   (254),A       ; вывести A в порт 254
250             INC   L             ; L=L+1
260             JR    NZ,BEEP       ; если L<>0, то перейти на BEEP
270             DEC   H             ; H=H-1
280             JR    NZ,BEEP       ; если H<>0, то перейти на BEEP
290             EI                  ; разрешение прерываний
300             RET                 ; возврат
```

Exploding 1

```
10 :GROUP       LD    HL,0          ; HL=адрес ПЗУ
20              LD    A,(23624)     ; A=
30              RRA                 ; цвет
40              RRA                 ;   бор-
50              RRA                 ;     дюра
60              LD    D,A           ; D=A
70 *FRQ,LEN     LD    BC,38401      ; C=частота,B=длина
80 LOOP1        PUSH  BC            ; сохранение BC
90 +DURATION    LD    B,20          ; B=длительность
100 LOOP2       LD    A,(HL)        ; A=содержимое ячейки ПЗУ
110             AND   248           ; сброс битов бордюра
120             OR    D             ; установка битов бордюра
130             OUT   (254),A       ; вывод A в порт 254
140             PUSH  BC            ; сохранение BC
150             LD    B,C           ; B=частота
160 LOOP3       DJNZ  LOOP3         ; задержка
170             INC   HL            ; HL=HL+1
180             POP   BC            ; восстановление BC
```

```
190             DJNZ  LOOP2         ; цикл
200             POP   BC            ; восстановление BC
210 ?INCREASE   INC   C             ; увеличение задержки (DEC C)
220             DJNZ  LOOP1         ; цикл
230             RET                 ; возврат
```

Exploding 2

```
10              LD    A,(23624)     ; A=
20              RRA                 ; цвет
30              RRA                 ;  бор-
40              RRA                 ;    дюра
50              LD    L,A           ; L=A
60 *FRQ,LEN     LD    DE,12801      ; E=частота,D=длина
70 LOOP1        PUSH  DE            ; сохранить DE
80 LOOP2        LD    B,E           ; B=E
90 PAUSE        DJNZ  PAUSE         ; задержка
100             LD    A,(BC)        ; A=содержимое ячейки ПЗУ
110             AND   248           ; сбросить биты бордюра
120             OR    L             ; установить цвет бордюра
130             OUT   (254),A       ; вывод A в порт 254
140             INC   C             ; увеличение адреса ПЗУ
150             INC   D             ; D=D+1
160             JR    NZ,LOOP2      ; если D<>0, то цикл
170             POP   DE            ; восстановить DE
180 ?INCREASE   INC   E             ; увеличение задержки (DEC E)
190             DEC   D             ; D=D-1
200             JR    NZ,LOOP1      ; если D<>0, то цикл
210             RET                 ; возврат
```

Volume FX

```
10              DI                  ; запрет прерываний
20 *FRQ,VOL     LD    BC,12900      ; C=частота,B=громкость
30 *LEN,DUR     LD    DE,25650      ; E=длина,D=длительность
40              LD    A,(23624)     ; A=
50              RRA                 ; цвет
60              RRA                 ;  бор-
70              RRA                 ;     дюра
80 LOOP1        PUSH  DE            ; сохранение DE
90 LOOP2        PUSH  BC            ; сохранение BC
100             XOR   16            ; инвертирование бита D4
110             OUT   (254),A       ; вывод A в порт 254
120 LOOP3       DJNZ  LOOP3         ; задержка (громкость)
130             XOR   16            ; инвертирование бита D4
140             OUT   (254),A       ; вывод A в порт 254
150             LD    B,C           ; B=C
160 LOOP4       DJNZ  LOOP4         ; задержка (частота)
170             POP   BC            ; восстановление BC
180             DEC   D             ; D=D-1
190             JR    NZ,LOOP2      ; если D<>0, то цикл
200             POP   DE            ; восстановление DE
210 ?DR:INC     INC   D             ; увеличение длительности (DEC D,NOP)
220 ?VL:DEC     DEC   B             ; уменьшение громкости (INC B,NOP)
230 ?FQ:INC     INC   C             ; увеличение частоты (DEC C,NOP)
240             DEC   E             ; E=E-1
250             JR    NZ,LOOP1      ; если E<>0, то цикл
260             EI                  ; разрешение прерываний
270             RET                 ; возврат
```

Flowing 1

```
10              DI                  ; запрет прерываний
20              LD    A,(23624)     ; A=
30              RRA                 ; цвет
```

```
40              RRA                  ;  бор-
50              RRA                  ;     дюра
60 *FRQ,LEN     LD    BC,65281       ; C=частота,B=длина
70 LOOP1        PUSH  BC             ; сохранение BC
80 +DURATION  LD      B,20           ; B=длительность
90 LOOP2        XOR   16             ; инвертирование бита D4
100             OUT   (254),A        ; вывод A в порт 254
110             PUSH  BC             ; сохранение BC
120             LD    B,C            ; B=C
130 LOOP3       DJNZ  LOOP3          ; задержка
140             POP   BC             ; восстановление BC
150             DJNZ  LOOP2          ; цикл
160             POP   BC             ; восстановление BC
170 ?DECREASE   DEC   C              ; уменьшение частоты (INC C)
180             DJNZ  LOOP1          ; цикл
190             EI                   ; разрешение прерываний
200             RET                  ; возврат
```

Flowing 2

```
10              DI                   ; запрет прерываний
20 *FRQ1,FRQ2   LD    DE,2660        ; E=частота 1,D=частота 2
30 +DURATION    LD    C,255          ; C=длительность
40              LD    A,(23624)      ; A=
50              RRA                  ; цвет
60              RRA                  ;  бор-
70              RRA                  ;    дюра
80 LOOP1        XOR   16             ; инвертирование бита D4
90              OUT   (254),A        ; вывод A в порт 254
100             LD    B,D            ; B=D
110 LOOP2       DJNZ  LOOP2          ; задержка 1
120             XOR   16             ; инвертирование бита D4
130             OUT   (254),A        ; вывод A в порт 254
140             LD    B,E            ; B=E
150 LOOP3       DJNZ  LOOP3          ; задержка
160 ?F1:INC     INC   D              ; увеличение задержки 1 (DEC D, NOP)
170 ?F2:DEC     DEC   E              ; увеличение задержки 2 (INC E, NOP)
180             DEC   C              ; C=C-1
190             JR    NZ,LOOP1       ; если C<>0, то цикл
200             EI                   ; разрешение прерываний
210             RET                  ; возврат
```

Cycle 1

```
10 +QUANTITY    LD    B,24           ; B=количество нот
20 :FREQUENCY   LD    HL,200         ; HL=начальная частота
30 :DURATION    LD    DE,1           ; DE=длительность
40              PUSH  HL             ; сохранение HL
50              PUSH  BC             ; сохранение BC
60              CALL  949            ; вызов подпрограммы ПЗУ
70              POP   BC             ; восстановление BC
80              POP   HL             ; восстановление HL
90 :STEP        LD    DE,30          ; DE=шаг изменения частоты
100 ?INCREASE   ADC   HL,DE          ; увеличение HL (SBC HL,DE)
110             DJNZ  :DURATION      ; цикл
120             RET                  ; возврат
```

Cycle 2

```
10 *STP,QUANT   LD    BC,12870       ; C=шаг,B=количество нот
20 :FREQUENCY   LD    HL,512         ; HL=начальная частота
30 :DURATION    LD    DE,20          ; DE=длительность
40              PUSH  HL             ; сохранение HL
50              PUSH  BC             ; сохранение BC
60              CALL  949            ; вызов подпрограммы ПЗУ
```

```
70              POP   BC            ; восстановление BC
80              POP   HL            ; восстановление HL
90              LD    A,L           ; умень-
100 ?DECREASE   SUB   C             ; шение (ADD A,C)
110             LD    L,A           ; L
120             DJNZ  :DURATION     ; цикл
130             RET                 ; возврат
```

Noise 1

```
10 :GROUP       LD    HL,0          ; HL=адрес ПЗУ
20              LD    A,(23624)     ; A=
30              RRA                 ; цвет
40              RRA                 ;  бор-
50              RRA                 ;    дюра
60              LD    D,A           ; D=A
70 :LENGTH      LD    BC,2560       ; BC=длительность
80 BEGIN        PUSH  BC            ; сохранение BC
90              LD    A,(HL)        ; A=содержимое ячейки ПЗУ
100             AND   248           ; сброс битов бордюра
110             OR    D             ; установка цвета бордюра
120             OUT   (254),A       ; вывод A в порт 254
130 +FREQUENCY  LD    B,40          ; B=частота
140 LOOP        DJNZ  LOOP          ; задержка
150 ?BOUNDS     INC   HL            ; HL=HL+1 (INC L)
160             POP   BC            ; восстановление BC
170             DEC   BC            ; BC=BC-1
180             LD    A,B           ; BC=
190             OR    C             ; 0 ?
200             JR    NZ,BEGIN      ; если нет, то цикл
210             RET                 ; возврат
```

Noise 2

```
10 :GROUP       LD    HL,0          ; HL=адрес ПЗУ
20 *FRQ1,FRQ2   LD    DE,2660       ; E=задержка 1,D=задержка 2
30 +DURATION    LD    C,255         ; C=длительность
40              LD    A,(23624)     ; A=
50              RRA                 ; цвет
60              RRA                 ;  бор-
70              RRA                 ;    дюра
80 BEGIN        XOR   16            ; инвертирование бита D4
90              OUT   (254),A       ; вывод A в порт 254
100             LD    B,(HL)        ; B=содержимое ячейки ПЗУ
110 LOOP1       DJNZ  LOOP1         ; задержка 1
120             LD    B,D           ; B=D
130 LOOP2       DJNZ  LOOP2         ; задержка 2
140             XOR   16            ; инвертирование бита D4
150             OUT   (254),A       ; вывод A в порт 254
160             LD    B,(HL)        ; B=содержимое ячейки ПЗУ
170 LOOP3       DJNZ  LOOP3         ; задержка 3
180             LD    B,E           ; B=E
190 LOOP4       DJNZ  LOOP4         ; задержка 4
200             INC   HL            ; HL=HL+1
210 ?F1:INC     INC   D             ; увеличение D (DEC D,NOP)
220 ?F2:DEC     DEC   E             ; уменьшение E (INC E,NOP)
230             DEC   C             ; C=C-1
240             JR    NZ,BEGIN      ; если C<>0, то цикл
250             RET                 ; возврат
```

Приложение 2.

Советы по использованию ассемблера.

Чтобы испытать эффекты в машинных кодах, приведенные в этой книге, Вам придется использовать специальную программу - ассемблер. В настоящее время существует множество различных ассемблеров, но в России наиболее популярен и распространен GENS4. Это составная часть пакета DEVPAC4 фирмы Hisoft. Вторая часть этого пакета (монитор - отладчик MONS4) нас сейчас не интересует.

Я не буду подробно описывать GENS4, это отлично сделано в [1], но некоторые ключевые моменты я объясню.

Существует несколько различных версий этой программы (в т. ч. и дисковых), поэтому ничего конкретного по загрузке посоветовать не могу кроме того, что для безопасности Вы можете пропустить бейсик-загрузчик и грузить сразу кодовую часть. Для её загрузки с кассеты введите следующие команды:

```
CLEAR 24999: LOAD "" CODE 25000
```

а с диска - такие:

```
CLEAR 24999: RANDOMIZE USR 15619: REM: LOAD "GENS4D" CODE 25000
```

Для запуска загруженного таким образом GENS4 введите команду

```
RANDOMIZE USR 25000.
```

Практически все эффекты, приведенные в этой книге можно ввести и ассемблировать без изменений. Исключение составляют листинги эффектов SUPER SOUNDa, в которых необходимо специальные метки заменить на более приемлемые (они должны начинаться с латинской буквы, быть несколько короче и не содержать пробелов).

Перед любым эффектом необходимо вставить директиву ассемблера ORG, указывающую с какого адреса разместить оттранслированный код. В данном случае этот адрес должен быть не меньше 37000. Например:

```
10            ORG 40000
```

Если Вы хотите вызывать созданный эффект прямо из ассемблера, то необходимо вставить директиву ENT:

```
20            ENT $
```

Теперь немного о командах строчного редактора GENS 4 (полный их список Вы можете найти в [1]):

I[N][,M]_	- Автоматическая нумерация строк начиная со строки N и с шагом M. Отмена - клавиша Edit (CS/1). По умолчанию N и M равны 10.
L[N] [,M]	- Вывод на экран листинга программы со строки N по строку M включительно. По умолчанию выводится листинг всей программы.
D N, M	- Удалить строки от N до M включительно.
G[,,S]	- Загрузить с магнитного носителя программу с именем S.
P[N][,M][,S]	- Записать на магнитный носитель программу со строки N по строку M включительно под именем S. По умолчанию используются параметры, установленные предыдущей командой G или P.
O[,,S]	- Записать на магнитный носитель откомпилированную программу под именем S.
R	- Запустить откомпилированную программу. Возврат в GENS4 осуществляется по команде RET.
B	- Вернуться в бейсик. Перезапустить GENS4 в нашем случае можно командой RANDOMIZE USR 25000.
Z	- Удалить текст программы.
H	- Вывести на экран список команд редактора.
A	- Ассемблировать программу. Формат этой команды довольносложен.

Подробно он описан в [1]. Здесь же Вам достаточно просто ввести букву A без параметров.

Список литературы.

1. Ларченко А.А., Родионов Н.Ю. ZX-Spectram и TR-DOS для пользователей и программистов - 3-е изд. СПб.: Питер, 1994.
2. Диалекты Бейсика для ZX-Spectram. Под редакцией Ларченко А.А., Родионова Н.Ю. СПб.: Питер, 1992.
3. АН СССР, и-нт проблем информатики. Лущихина И.Ю. и др. Организация цифрового музыкального интерфейса MIDI. Москва: Препринт, 1988.
4. MIDI Musical Instrument Digital Interface Specification 1.0 North Hollywood: International MIDI Association (IMA), 1983.
5. Schulze, Hans-Jochen/Engel, Georg. Moderne Musikelektronik, Praxisorientiere Elektroakustik und Gerate zur elektronischen Klangerzeugung. Berlin: Militarverlag der DDR (VEB), 1989.